Wissenschaftsgenese und Wissenschaftskonzepte der informatischen Disziplin Computerarchitektur: Modelle der Artefaktstruktur, des Designraums und der Designkonflikte ; Bericht des Theorieprojektes

Hellige, Hans Dieter

artec Forschungszentrum Nachhaltigkeit

Hans Dieter Hellige

Wissenschaftsgenese und Wissenschaftskonzepte der informatischen Disziplin Computerarchitektur: Modelle der Artefaktstruktur, des Designraums und der Designkonflikte. Bericht des Theorieprojektes

artec-paper Nr. 99
November 2002

ISSN 1613-4907

Universität Bremen

artec | Forschungszentrum Nachhaltigkeit
Enrique-Schmidt-Str. 7
Postfach 330 440
28334 Bremen
http://www.artec.uni-bremen.de

Wissenschaftsgenese und Wissenschaftskonzepte der informatischen Disziplin Computerarchitektur: Modelle der Artefaktstruktur, des Designraums und der Designkonflikte
Bericht des Theorieprojektes

Hans Dieter Hellige

Wissenschaftsgenese und Wissenschaftskonzepte der informatischen Disziplin Computerarchitektur: Modelle der Artefaktstruktur, des Designraums und der Designkonflikte, Bericht des Theorieprojektes

Hans Dieter Hellige

"Der entscheidende Fehler der ersten Versuche zur Erreichung einer Wissenschaft des Konstruierens bestand darin, daß diese nach dem Vorbild bereits vorhandener, schon weiter fortgeschrittener Wissenschaften geschaffen werden sollte. Die ganz eigenartige Vielseitigkeit des Konstruierens läßt dieses Bestreben heute als grundsätzlich falsch erkennen. Das Konstruieren kann als eine Geistestätigkeit ureigenen Gepräges nur nach eigenen Gesetzen gelehrt werden."

Hugo Wögerbauer 1943[1]

0. Zum Forschungsstand der Informatikgeschichte

Bislang kann von einer *Geschichte der Informatik* noch keine Rede sein, sondern besten-falls von *Geschichten der Informatik*. Der Plural verweist dabei zum einem auf die vielfach nur in Form von Geschichtserzählungen beteiligter Pioniere bestehenden Darstellungen. Zum anderen ist er Ausdruck der noch immer nicht zum Abschluß gekommenen Disziplinwerdung des Faches wie des unfertigen Standes der wissen-schafts- und technikhistorischen Aufarbeitung, die eine Gesamtdarstellung *der* Geschichte *des* Fachs noch nicht zulassen. Darüber hinaus betont der Plural ‚Geschichten' die Perspektivgebundenheit historischer Rekonstruktionen und korrespondiert insofern mit der Aufsatzsammlung „Sichtweisen der Informatik"(Coy u.a. 1992), in der der Titel anstelle des zu anspruchsvollen Labels „Theorie der Informatik" gewählt wurde. Denn so wie alle bisherigen Versuche, die Informatik als ein geschlossenes Theoriegebäude auf der Grundlage ‚letzter Theorien' zu begründen gescheitert sind und von einer Vielzahl theoretischer Perspektiven abgelöst wurden, so wenig erscheint es derzeit oder überhaupt möglich, ein geschlossenes Bild der Informatik-Geschichte auf der Basis eines anerkannten Entwicklungsmodells zu entwerfen.

---

[1] Wögerbauer, H., Die Technik des Konstruierens, 2. Aufl. Berlin 1943, S. 165

Verschärfend kommt hinzu, daß die historische Erforschung der Disziplinentwicklung noch immer im Schatten der maschinenfixierten Computergeschichtsschreibung steht. Trotz der erfreulichen Ausweitung der Forschungsgegenstände und Fragestellungen im letzten Jahrzehnt bilden Rechnerhistorien, herausragende Pioniergestalten und Gründerunternehmen noch immer den Schwerpunkt geschichtswissenschaftlicher Studien. Im Unterschied zu der theoretisch-methodisch fortgeschritteneren Geschichte der Elektrotechnik und der Telekommunikation geht es hier weitaus mehr um Prioritätsansprüche, Patentstreite und Erfinderbiographien in traditionell heroischer, narrativer Darstellungsweise. Die Wissenschaftsgeschichte der Informatik hat demgegenüber eine viel geringere Aufmerksamkeit gefunden. Es gibt zwar erste Überblicksdarstellungen zur Institutionalisierung und Professionalisierung des Faches und wenige eher spekulative Gesamtsynthesen auf philosophischem bzw. wissenschaftssoziologischem Hintergrund. Doch es fehlen Darstellungen der Genese dee Wissenschaftskonzepte und der Entwicklung des Fächerkanons der Computer Science bzw. Informatik. Auch die Teildisziplinen sind bislang sehr ungleichmäßig erforscht, so daß ihr wechselseitiger Zusammenhang nur in Umrissen erkennbar wird.

Zur fehlenden empirischen Aufarbeitung kommen wissenschaftssoziologische Theorieprobleme hinzu: Die Informatik entspricht nur bedingt den üblichen Modellen der Verwissenschaftlichung, Paradigmenbildung und Entwicklungsstufen, wie sie seit Kuhn, Lakatos meist anhand von Naturwissenschaften oder naturwissenschaftsnahen Technikwissenschaften erstellt wurden. Dies liegt nicht zuletzt darin begründet, daß dieses Fach durchweg heterogener ist und von Beginn an in viel höherem Maße als eine interdisziplinäre Wissenschaft angelegt war. Howard Aiken hat dieses schon 1956 bündig formuliert: „Our difficulty is further that data processing and all other automatic control processes are by their very nature interdisciplinary in character." (Aiken 1956, S. 34). Es empfehlen sich deshalb eher von Entwicklungsstufen-und Paradigmenkonzepten losgelöste Analysemodelle, wie sie etwa Michael Mahoney (1998/2000) mit seinem „Agenda"-Ansatz für die Mathematik- und Informatikgeschichte vorgeschlagen hat. Doch gleichgültig mit welchem theoretischen Modellen oder narrativen Methoden der Wissenschafts- und Technikhistoriker arbeitet, er stößt bei der Informatik auf Unübersichtlichkeiten und Schwierigkeiten, die es in diesem Ausmaß in klassischen Ingenieurwissenschaften nicht gibt. Denn seit Beginn der Institutionalisierung der Computer Science ab 1960 in den USA und ab 1965 in Europa ist die Debatte über Gegenstandsbereich, Wissenschafts-

vorbilder, Wissenschaftscharakter und die Einordnung in die Wissenschaftssystematik nicht zur Ruhe gekommen.

Auch lange nach Abschluß der Professionalisierung und Institutionalisierung ist die wissenschaftliche Begründung und Theoriebildung noch nicht abgeschlossen. Folglich ziehen sich die Feststellungen bzw. Klagen über den unfertigen Zustand der Disziplin bis heute durch die unzähligen Positions- bzw. Entschließungspapiere und die schon zum Topos gewordenen Selbstverständigungstexte „Was ist Informatik". So beantworteten Newell, Perlis und Simon 1967 die Frage „What is Computer Science?" nicht mit einer bündigen Definition, sondern mit einem ergebnis-offenen Sokratischen Dialog. Ein Jahr später resümiert Richard W. Hamming: „ ‚What is copmputer science?' is argued endlessly among people in the field". Heinz Zemanek sah 1971 in „Informatik" ein gut gewähltes Wort, dem der Begriff noch fehlt, und deutete bereits an, daß mit einer ‚endgültigen' Definition ohnehin nicht zu rechnen sei. In der Bestandsaufnahme von 1982 kommt Seymour V. Pollack zu dem Resultat: „There is still no ‚standard' (i.e. universally inoffensive) definition of computer science. In fact, the existence of such a discipline continues to be a debatable point for a substantial number of people."(S. VII f.). In den achtziger und neunziger Jahren verschärfen sich die Kontroversen um die Positionierung und Strukturierung der Disziplin sogar noch, wie es auch die zahl-reichen wissenschaftstheoretischen, fachpolitischen und curricularen Diskussionspapiere im Informatik-Spektrum und in den Communications of the ACM belegen. Die Tatsache der auch im 42. Jahr der ACM noch ungeklärten Frage, ob „Computer Science" eine Wissenschaft, eine Ingenieurdisziplin oder eine bloße Technik ist, bewog eine Task Force der ACM unter Leitung von Peter Denning dazu, die Kerndisziplin endlich zu definieren (Denning, Comer, Gries). Doch spätere Grundsatzartikel Dennings zeigen allzu deutlich, daß der 1989 vereinbarte Kanon inzwischen mehrfach revidiert werden mußte. Und selbst jüngste kritische Bestandsaufnahmen kommen noch zu dem Resultat, daß die Informatik „still under Construction" sei (Coy 1997).

Die Unabgeschlossenheit der Wissenschaftswerdung wird dabei teils positiv als Ausdruck der flexiblen Anpassung an neue Anforderungen bewertet, teils aber auch als Hinterherhinken der Academia hinter der Praxis und als unsystematische Muddling-through-Mentalität negativ beurteilt. Für Jörg Desel (2001, S. V) ist die Informatik auch nach drei bis vier Jahrzehnten noch so jung, daß sich weder „eine Aufteilung in Bereiche mit einem Grundstock an Inhalten noch eine informatikeigene Denk- und Vorgehensweise fest etabliert hätten".

Entwicklungsmuster von (Technik-/Natur-) Wissenschaften

(Stufenkonzept in Anlehnung an D. v. Engelhardt)

Wissenschaftsexterne Entwicklungsstufen:

Professionalisierung:
Bildung einer Spezialistengruppe, Berufsgruppe, eines Berufsverbandes

Institutionalisierung:
Herausbildung eines Spezialgebietes innerhalb einer etablierten Wissenschaft, regelmäßiges Lehrangebot, Entstehen einer Teildisziplin, einer neuen Fachdisziplin durch Integration von Forschungs- und Lehrgebieten aus anderen Disziplinen
Schaffung der Disziplinbezeichnung als ‚Markenzeichen'
Interne Fächerdifferenzierung und Ausbildung eines Fächerkanons
Bildung von Fachschaften, Angleichung lokaler Curricula zu (inter-)national anerkannten Studiengängen, Entstehen von (inter-)nationale Konferenzen, Fachgesellschaften, Fachverbänden

Literalisierung:
Fachbücher, Lehrbücher, Fachzeitschriften

Wissenschaftsinterne Entwicklungsstufen:

Grundlegung/Autonomisierung:
Etablierung von Fachdiskursen, Loslösung von Mutterdisziplin(en) und deren Theoriemustern und Methodenidealen, Theorie- / Methodenstreite vor und nach Etablierung einer Disziplin

Methodisierung:
Herausbildung eines fachspezifischen Methodenspektrums, Fachsprachenbildung

Empirisierung:
Systematisierte Erfahrungsakkumulation, vergleichende Struktur- und Bauformenlehre, Erfahrungswissenschaft

Systematisierung, Theoretische Fundierung, Kanonisierung
Richtungsstreite zwischen theoretischer und empirischer Orientierung
Paradigmatische Schließung
Entstehen eines Theoriegebäudes

Probleme und Grenzen szientistischer Stufenkonzepte bei der Informatik

Implizierte Entwicklungslogik der Stufenmodelle:

Kunstlehre, systematisierte Erfahrungsakkumulation, Erfahrungswissenschaft, formalisierte systematische Wissenschaftsdisziplin, paradigmatische Schließung (Kuhn)
Empirische Basis der Entwicklungslogik: Naturwissenschaften, naturwissenschaftsnahe Technikwissenschaften

Abweichen der Informatik vom klassischen Verwissenschaftlichungsmodell: Rücknahme paradigmatischer Schließungen infolge permanenter Änderungen der Rahmenbedingungen und Aufgaben, Nutzungsabhängigkeit und Nutzerspezifik

Mischcharakter des Wissenschaftstyps der Informatik

Mathematische und formaltheoretische Grundlagendisziplinen:

Automatentheorie, Berechenbarkeitstheorie, Formale Sprachen, Rekursionstheorie, Komplexitätstheorie, Informationstheorie, Codierungstheorie

Designorientierte bzw. konstruktive Teildisziplinen:

Rechnerarchitektur, Softwarekonstruktion / Softwarearchitektur, Netzwerkarchitektur, Mensch-Computer-Interface-Design Mediengestaltung

Anwendungsbereich-bezogene Teildisziplinen:

Bindestrich-Informatiken, Erfordernis anderer Theoriekonzepte und nicht-mathematischer Modellbildung, Erfordernis der Einbeziehung sozialer und hermeneutischer Prozesse, Stakeholder-Design-Konzepte

Probleme szientistischer Entwicklungsmodelle

- Problematik der strikten Modellierung einer Disziplin nach einer anderen,
- meist strengeren Technik- oder Naturwissenschaft
- Fixierung auf Theoriemuster und Methodenideale von Leitdisziplinen führt zur
- Ausblendung spezifischer Probleme konstruktiver Disziplinen:
  Problematik der Komplexität, Perspektivität und Zielkonflikthaftigkeit

Der Vorteil der innovativen Flexibilität in der Vergangenheit könne sich jedoch in Zukunft als Positionsschwäche desintegrierend auf die Disziplin auswirken. Hans-Willy Hohn kommt in seiner wissenschaftssoziologischen Studie 1998 (S. 136) gar zu dem kritischen Resümee, daß die Informatik vom „Status einer Normal-wissenschaft“ weit entfernt und ihre Identität durch Ausweitung des Anwen-dungs-spektrums heute umstrittener denn je sei. Mit ganz anderer Wertung gelangt Peter Rechenberg (2000, S. 300) zum selben Ergebnis: „Mir scheint, daß wir, von den theoretischen Grundlagen her betrachtet, noch gar nicht wissen, was eigentlich der Kern der Informatik, die Essenz von Computern ist.“ Erst wenn das Computermodell der Theoretischen Informatik, die Turingmaschine, von einer richtigeren Abstraktion überwunden würde, „könnte die Informatik als eigenständige Wissenschaft“ klarer hervortreten. Daß dies bisher nicht möglich war, daß eine derartige Vielfalt der Blickwinkel und Definitionen herrsche, ist für ihn wie für viele andere Fachvertreter aber der Beweis für eine sehr lebendige, sich noch entwickelnde Wissenschaft. Der vorparadigmatische Zustand der Disziplin wird damit zum Beleg ihres noch ungebrochenen Innovationspotentials. So muß die Informatik mit dem Widerspruch leben, daß sie mit Blick auf die unangefochtene Stellung des Computers als Schlüsseltechnik die Rolle einer Leitdisziplin des 21. Jahrhunderts beansprucht, zugleich aber noch immer keine allgemein anerkannte Bestimmung ihres Gegenstands, ihres Wissenschaftstyps und ihres Wissenschaftskerns vorweisen kann.

Historiker können mit ihren wissenschaft-lichen Mitteln natürlich nun nicht entscheiden, welche Definition des Faches richtig und welche Position in dem wissenschaftspolitischen Dauerstreit angemessener ist. Mit ihrer diachronen Perspektive können sie aber langfristige Entwicklungstrends, disziplinäre Beharrungstendenzen, Wertwanderungen sowie soziale und kulturelle Kontexte besser sichtbar machen, als es die synchrone Betrachtung des jeweiligen Richtungsstreites vermag. So wird aus der Langzeitperspektive das Zusammenwirken von charakteristischen Prozessen bei der Konstituierung eines neuen Faches, der technisch-wissenschaftlichen Erfahrungsakkumulation und der Ausweitung von Anwendungsfeldern und gesellschaftlichem Wirkungsspektrum besser sichtbar.

Die Frühphase stand ganz im Zeichen des Kampfes der Ursprungsdisziplinen technische bzw. angewandte Mathematik und Nachrichtentechnik um die Einverleibung des neuen Gebietes in den jeweiligen Wissenschaftsbereich. Für Alwin Walther entwickelte sich die Rechentechnik zur allgemeinen Informa-

tionsverarbeitung, führe über diese zu einer Wiedervereinigung der „Gesamtmathematik“ und verwirkliche so die „mathesis universalis“. Für Maurice Wilkes und Karl Steinbuch waren dagegen die Nachrichtentechnik bzw. die elektrische Nachrichtenverarbeitung die natürliche Heimat des neuen Gebietes. Im Tauziehen zwischen diesen beiden Communities gewann das neue Gebiet immer mehr Eigenständigkeit, doch wurde es noch lange als bloße Schnittmenge definiert, es blieb eine Grenzdisziplin die noch dazu den Gegensatz der Wissenschaftstypen Ingenieur-/Natur- und Geisteswissenschaften geerbt hatte: „Computer science is a blend of engineering and mathematics. Being comparatively new, the field has not settled down to a definite mixture of science and engineering but still contains large parts of both technology and art.“ (Hamming 1971, S. IX) .

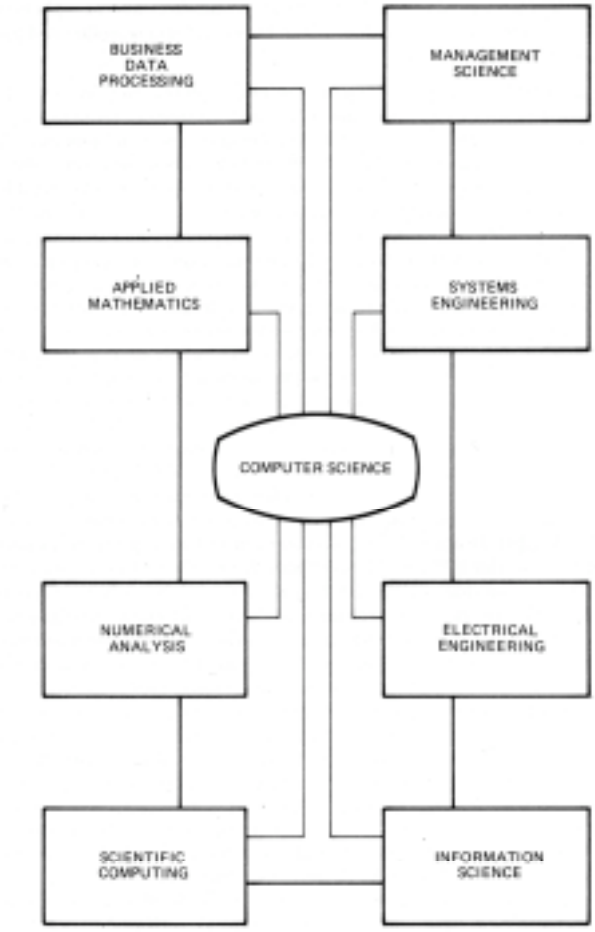
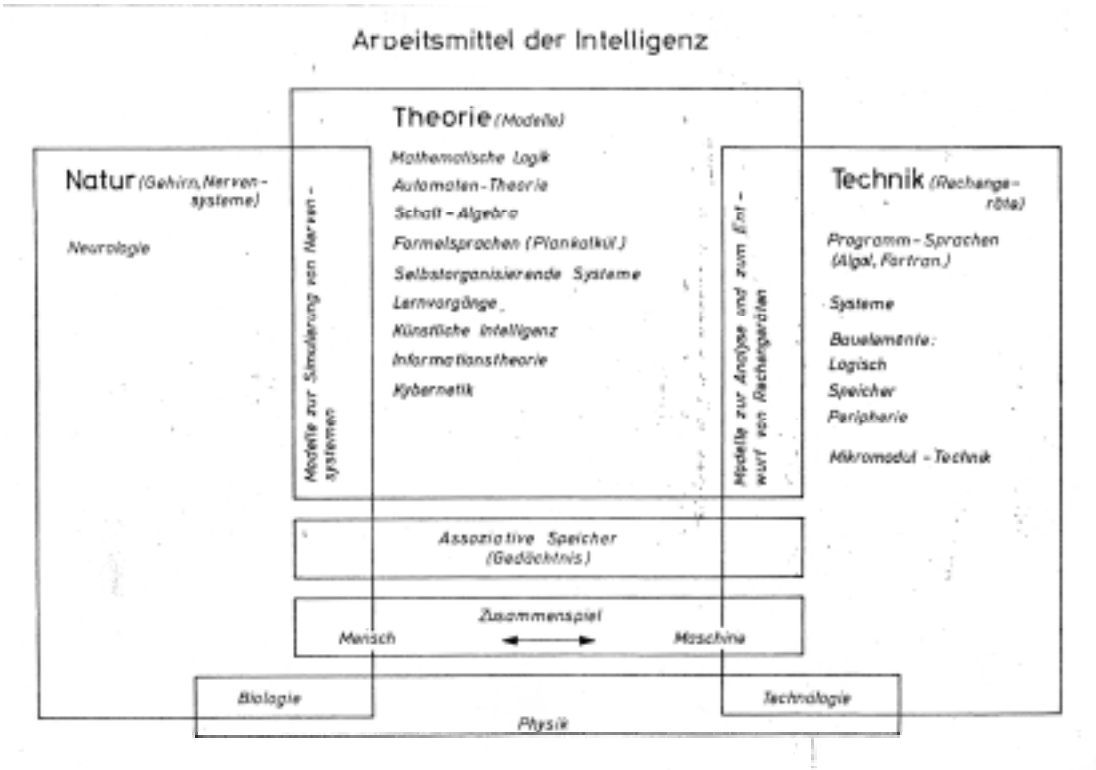
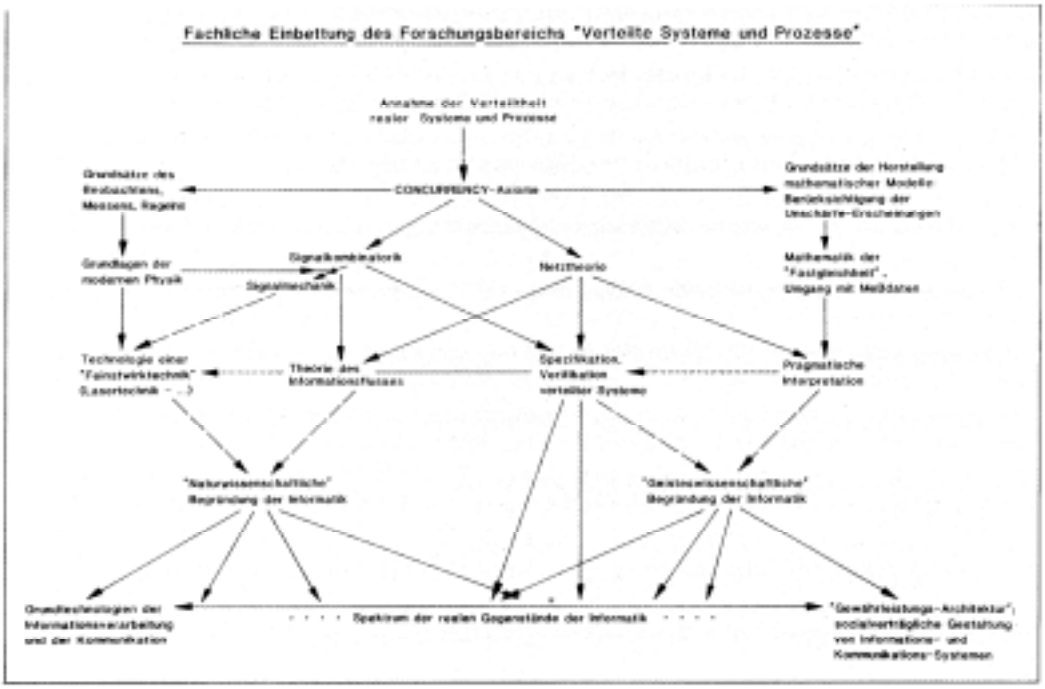


Abb. 1 Einordnung der Computer Science in dem Lehrbuch von Richard C. Dorf von 1972 (S. 4)

Die trotz fortschreitender Autonomisierung weiter bestehenden Legitimationsdefizite wurden bald Anlaß zu Anleihen bei allgemeinsten Begründungsmustern

etablierter Naturwissenschaften. Vor allem die Physik war neben der Mathematik als Leitdisziplin besonders attraktiv, da sie den Gesetzescharakter des theoretischen Kerns mit dem Universalitätsanspruch verband.





Aufbau der Informatik nach dem Vorbild der exakten Wissenschaften und die Heranziehung der Naturgesetzlichkeiten des Informationsflusses. aus: GMD-Jahresbericht 1984

Abb. 2 Die disziplinäre Verortung der Informatik bei Konrad Zuse und Carl Adam Petri (Zuse 1965; Petri, Schaubild in: GMD Jahresbericht 1984 (nach Bolkart, Haug 1986, S. 16)

So gründete Konrad Zuse um 1965 das neue Gebiet unter der Bezeichnung „Arbeitsmittel der Intelligenz“ auf einem Fundament aus Logik, Automatentheorie, Formelsprachen, selbstorganisierende Systeme, Künstliche Intelligenz, Informationstheorie und Kybernetik und verknüpfte dieses mit der Physik, Biologie und Technologie zu einer neuen Überwissenschaft. Carl Adam Petri deutete asynchrone Informationsflüsse relativitätstheoretisch und integrierte auf diese Weise die Kommunikations- und Informationsprozesse in das anerkannte Theoriegebäude der Physik ein. Karl Ganzhorn (1973) schließlich sah in der Äquivalenz zwischen Steuerung und Information bei der Programmspeicherung eine Entsprechung zu den grundlegenden Äquivalenzen der Thermodynamik und ließ die Informationsverarbeitung so mit einem Schlage in den Olymp physikalischer Naturgesetze aufrücken.

Auch die in den ersten Jahrzehnten häufiger Wahl der Fundamentalkategorie „Information“ als zentralen Gegenstand der Disziplin diente letztlich der Schaffung eines unanfechtbaren Platzes im Wissenschaftssystem. Alan Perlis brachte dies 1968 auf den Punkt: „Computer science has one goal: the understanding of the organization and administration of information. This goal is as fundamental as in physics is the search for the key to the organization of matter and energy.“ Es hat den Anschein, daß selbst der erste Vorschlag des Begriffes „informatique“ als Disziplinbezeichnung in der Sitzung der „l’Association Française de Calcul et Traitement de l’Information“ im Jahre 1962 von Philippe Dreyfus ursprünglich als sprachliche und institutionelle Mimikry gedacht war, als bewußte Anlehnung an die bereits etablierten Wissenschaften der „mathématique“ und „physique“, und nicht, wie meist angenommen, als eine Synthese von „information’ und „automatique“ (Arsac 1970, S. 48)

Informatik als Leitdisziplin?

Begriff Leitdisziplin: Das Paradigma, die Kerntheorien und die zentrale wissenschaftliche Methode übt einen prägenden Einfluß auf die anderen technisch-naturwissenschaftlichen und z. T. sogar geistes- und sozialwissenschaftlichen Disziplinen aus.

In der Neuzeit waren vor allem die Physik und hier besonders die Mechanik mit der zugehörigen Mathematik Leitdisziplin aller Technik- und Naturwissenschaften. Alle neuen Gebiete, wie die Elektrotechnik und Nachrichtentechnik, folgten zunächst dem Theorie- und Methodenspektrum der Physik. Im 19. Jahrhundert erweiterte sich die Mechanik zur Thermodynamik, die alle thermischen und energetischen Prozesse in einer das ‚Weltschicksal' bestimmenden Theorie bündelte. In der 2. Hälfte des 19. Jahrhunderts rückte über die Evolutionstheorie die Biologie mehr und mehr ins Zentrum, sie prägte auch organologische, vitalistische und sozialdarwinistische Sichtweisen von Politik und Gesellschaft – der Staatsorganismus löste die mechanistische Staatsmaschine ab. Seit Mitte des 20. Jahrhunderts traten Nachrichtenverarbeitung, Informationstheorie, Informationswissenschaft und vor allem Informatik auf den Plan, um die bislang stoff- und energiebezogenen Leitdisziplinen abzulösen: Die Information als neue Fundamentalkategorie hielt bald, vermittelt über Computer als zentrale Modellierungswerkzeuge in alle Wissenschaften Einzug. Selbst Gene werden seitdem als Informationsträger gesehen und soziale Strukturen als informationelle Netzwerke interpretiert. Auf die Agrar-, Industrie- und Dienstleistungsgesellschaft folgt nun die „Informationsgesellschaft" als neue bestimmende Gesellschaftsformation. Die Informatik bzw. Computer Science sieht sich seit den 60er Jahren als das wissenschaftliche Zentrum und die neue Leitdisziplin. Obwohl vergleichsweise noch sehr jung, rückte sie über eine derartige wissenschaftstheoretische und universalhistorische Verankerung gleich auf Augenhöhe mit den alten Leitdisziplinen Physik/Mathematik und Biologie. Im Gegensatz zu diesem Anspruch steht der Zustand der Disziplin: Sie ist auch nach über 40 Jahren noch immer weit entfernt vom Status einer intern und extern anerkannten Normaldisziplin. Der Theoriekern ist seit Jahrzehnten Gegenstand intensiver Debatten, der Fächerkanon hat sich gewohnheitsrechtlich durchgesetzt mit zum Teil erheblichen Abweichungen in verschiedenen Ländern. Struktur und Begründung des Fächerkanons sind derzeit wieder höchst umstritten, da das in den 60er/70er Jahren geschaffene Theoriegebäude kaum noch Verbindlichkeit besitzt. Sie kann somit noch keine anerkannte Bestimmung ihres zentralen Gegenstandes, ihres Wissenschaftstyps und ihres Wissenschaftskerns vorweisen.

- Ist sie eine **Grundlagendisziplin** wie die Mathematik oder Physik?
- Ist sie eine **Technikwissenschaft** um die Kernmaschine Computer herum?
- Wenn ja, was ist ein Computer: ein Rechenautomat, Universalautomat, eine Zeichen- /Symbolverarbeitende Maschine, ein universales Modellierungswerkzeug, ein Informationsmedium, eine in Alltagsgegenstände/-prozesse inkorporierte Steuerungs- und Programmierintelligenz?
- Ist sie eine **Wissenstechnik** oder die **Informationsdisziplin**, die alle informationellen Prozesse abwickelt und organisiert?
- Ist sie eine **Organisations-/Arbeits- oder Gesellschaftswissenschaft**, die die gesamtgesellschaftliche Arbeit über informationelle Arbeitsinstrumente und –systeme strukturiert und organisiert?
- Ist sie eine **Grenzdisziplin** zwischen verschiedenen Wissenschaftstypen?
- Was der **Kernbegriff der Informatik?** :
- Algorithmus/Berechnung
  Information/Kommunikation
  Symbol/Zeichen
  System
  Struktur
  Modellbildung
  Arbeit / Gesellschaft / Organisation

Diesen an Mathematik und Physik angelehnten theoretischen Begründungskonzepten wurde von Vertretern eines konstruktiv-ingenieurmäßigen Wissenschaftsverständnisses entgegengehalten, daß Informatiker es mit Artefakten und „hard facts of life“ zu tun hätten, die weit unter den allgemeinsten mathematischen Theoremen und Naturgesetzen angesiedelt seien (Hamming, 1971; Pollack 1982, S. VII). Pragmatische *Technik*-Orientierungen wurden begünstigt von der Expansion der Anwendungsfelder, dem Übergang zum „Interactive Computing“ sowie von der zunehmenden Werkzeug- und Medien-Funktion des Computers. Diese führten in immer neue Zeichen-, Symbol- und Modellräume, die wegen hoher Zeit- und Kontextabhängigkeit nicht mehr durchgängig berechenbar und algorithmisierbar waren. Das auf Algorithmen und Formalisierung ruhende theoretische Fundament erwies sich so als nicht mehr ausreichend. Die Informatik hatte zu lernen, die ungeordnete Welt der Anwendungen in die „Programmierungstheorie“ einzubeziehen und „die Spannung zwischen der logischen Welt der Ja-Nein-Entscheidungen und der wirklichen Welt der offenen und unklaren Entscheidungen (zu) überbrücken.“(Zemanek 1971, S. 160). Für Frederick Brooks lag das Hauptproblem der Disziplin in der Überwindung von Design-Komplexität und diese ließ sich weder durch abstrakte Modellierung bewältigen noch durch den Versuch, Heterogenität und Vielfalt auf eine ”fundamental unified theory” zurückzuführen. Er hielt deshalb letztlich sogar den Begriff der ”Computer Science” für eine irreführende, die Zunft fehlleitende Bezeichnung (Brooks 1977, S. 625).Es entstanden so nicht nur in den neuen anwendungsnahen Teilgebieten der Informatik wie der Wirtschaftsinformatik, der Ingenieurinformatik und der „Computer Education for Management“, sondern auch in frühen Curriculums-Konzepten für das System- und Software Engineering Überlegungen, neben der mathematisch-theoretischen „Computer Science“ eine *konstruktive, ingenieurmäßige* Fachrichtung zu etablieren, die die Praxisprobleme, Designkonflikte und Managementaspekte in die Ausbildung integriert. Doch die pragmatischen Orientierungen hatten es lange Zeit schwer, sich gegenüber den szientistischen Leitbildern und Universalansprüchen durchsetzen, erst seit den Neunzigern sind sie klar auf dem Vormarsch (siehe u.a. Broy, Schmidt 1999).

TABLE I

Recommended Curriculum for Computer Science Majors

Courses / Recommendations	Computer Science				Supporting
	Basic Courses	Theory Courses	Numerical Algorithms	Computer Models and Applications	
Required	1. Introduction to Algorithmic Processes 2. Computer Organization and Programming 4. Information Structures	5. Algorithmic Languages and Compilers	3. Numerical Calculus (or Course 7)		Beginning Analysis (12 cr.) Linear Algebra (3)
Highly Recommended Electives	6. Logic Design and Switching Theory 9. Computer and Programming Systems		7. Numerical Analysis I 8. Numerical Analysis II		Algebraic Structures Statistical Methods Differential Equations Advanced Calculus Physics (6 cr.)
Other Electives	10. Combinatorics and Graph Theory	13. Constructive Logic 14. Introduction to Automata Theory 15. Formal Languages		11. Systems Simulations 12. Mathematical Optimization Techniques 16. Heuristic Programming	Analog Computers Electronics Probability and Statistics Theory Linguistics Logic Philosophy and Philosophy of Science

Reprinted by permission from "An Undergraduate Program in Computer Science—Preliminary Recommendations," *Communications of the Association for Computing Machinery*, 8 (1965), 543–552, copyright 1965 by the Association for Computing Machinery.

Die seit der Mitte der sechziger Jahre in Europa auch unter der Bezeichnung „Informatique“ und „Informatik“ antretende Wissenschaft hatte, entgegen den diversen theoretischen Begründungsansätzen, vorwiegend Patchwork-Charakter. In dem disziplinären Konglomerat von „Grundwissenschaften“ bearbeitete sie die informationellen Fragen und Programmierungsaspekte und entwickelte daraus nach und nach eine eigenständige „Technikwissenschaft neuen Typs“ (Zemanek 1971, Zadeh 1969, Cardenas 1972, Bauer 1974). Die clusterartige Struktur, die sich auch sehr deutlich in den frühen Lehrbüchern und Lehrplänen spiegelte, führte aber bald zu Bemühungen um eine stringentere Aufteilung in Kern- und Randdisziplinen. In der amerikanischen und vor allem in der gezielt als eigenständige Disziplin konzipierten deutschen Informatik kam es 1968/69 zur ersten Abgrenzung von technischen Grundlagenfächern, „Informatik-Kerngebieten“ und „angewandter Informatik“. Dabei waren anfangs - entsprechend der Dreiteilung der Gebiete in Theorien, Systeme und Anwendungen in den ACM-Empfehlungen von 1968 - die „Technische Informatik“ mit der Programmierung und Systementwicklung noch in der „Systemorientierten Informatik“ zusammengefaßt. Ab 1972 bürgerte sich dann die Aufteilung in theoretische, praktische, technische und angewandte Informatik ein (Haacke, Fischbach, 1972, S. 8, 14 ff.; Donth 1984). Diese hielt sich, trotz mancher Veränderungen im theoretischen Überbau, nahezu unverändert bis zur Gegenwart, erst in den letzten Jahren deutet sich die Wiedervereinigung von

technischer und praktischer Informatik zu „Informatiksystemen“ und damit die Rückkehr zum Dreisäulenmodell an (Wilhelm, S. 12 f.; Mayr, Maas, S. 178)

INFORMATIK			
Technische	Praktische	Theoretische	Angewandte
Hardware-komponenten	Algorithmen, Datenstrukturen, Programmier-methoden	Automatentheorie	Computergrafik
Schaltnetze, Schaltwerke, Prozessoren	Programmier-sprachen und Übersetzer	Formale Sprachen	Datenbanken
Mikro-programmierung	Betriebssysteme	Theorie der Berechenbarkeit	Künstliche Intelligenz
Rechnerorganisation und -architektur	Softwaretechnik	Komplexitätstheorie	Digitale Signalverarbeitung
Rechnernetze	Mensch-Maschine-Kommunikation	Formale Semantik	Simulation und Modellierung
	Verteilte Systeme	Automatische Programmsynthese	Textverarbeitung und Büroautomatisierung
			Spezifische Anwendungen in Wirtschaft, Verwaltung, Ingenieurwissenschaften, Naturwissenschaften, Medizin, Geisteswissenschaften, Kunst

Bild 3 Eine Einteilung der Informatik

Nach dem Sortieren der Teildisziplinen Anfang der 70er Jahre entstand bald das Bedürfnis nach einer systematischen Ordnung des Disziplingefüges und einer Zusammenfassung der diversen Einzeltheorien, Erfahrungsregeln und formalen Methoden zu einem „geschlossenen theoretischen System“, ein Vorhaben, das die junge Informatikforschung als eine ihrer Hauptaufgaben ansah (Giloi 1970, S. 7). Die Strukturierung geschah vornehmlich in der Form hierarchischer Schichten- oder Schalenmodelle, die die Artefakte, Methoden und Theorien stufenartig vom ‚exakten‘ theoretischen Kern bis zur empirischen Ebene der Phänomene und der praktischen Anwendungen aufbauten. Perlis (1968, S. 70) wählte die Metapher des Zwiebelschalenmodells: „Computer science is the study of the design, analysis, representation, and application of computers. All aspects of this study are like the layering of the onion.“ Am häufigsten wurde aber das System der Informatik als ein Gebäude vorgestellt, wobei dem *theoretischen Fundament* von vornherein die tragende Schlüsselstellung zuerkannt wurde. Jürg Nievergelt (1995) stellte das Disziplingefüge später als „Informatik-Turm“ dar, bei dem jedes Stockwerk auf den Leistungen des jeweils darunter liegenden beruht: die Anwendungsmethodik auf der System-Realisierung, diese auf der Algorithmik und alle auf der abstrakten mathematischen Theorie, die mit ihren allgemeingültigen Grundgesetzen sowohl Fundament der Disziplin als auch letzte Quelle von Basisinnovationen ist.

Im Unterschied zur Elektrotechnik, wo die Elektrodynamik als theoretische Grundlage der Disziplin relativ unangefochten ist, konkurrierten in der Informatik von Beginn an immer verschiedene Theorien um den disziplinären Kern: so die als Gruppe auftretende Berechenbarkeitstheorie, Automatentheorie und die Formale Sprachen, dann die Theorie der Programmierung, die Informationstheorie, Kommunikationstheorie und die allgemeine Theorie der „Zeichenverarbeitung", schließlich die multidisziplinär angelegten Theoriekomplexe der Kybernetik und der „Automatique", d.h. der besonders in Frankreich angestrebten Schaffung eine integralen Theorie der mechanischen und informationellen Automaten und Automatismen (u.a. Deveaux). Lange Zeit konnten die mathematisch orientierten Disziplinen der „Theoretischen Informatik" den Wissenschaftskern erfolgreich behaupten. Seit den achtziger Jahren bricht jedoch das starre Gefüge immer mehr auf. Durch Workstations, PCs, Laptops, Notebooks, mobile und am Körper tragbare Computer wird Rechnerintelligenz zunehmend dezentralisiert und in Alltagsprozesse integriert. Vor allem im „Embedded" und „Ubiquitous Computing", rücken die Anwendungen, einst an der Peripherie plaziert, nun ins Zentrum. Die durchgängige Verknüpfung von Informatik-Systemen mit physikalischen Prozessen, technischen Geräten und Abläufen der Lebenswelt steigert die Bedeutung qualitativer Repräsentationen, hermeneutischer Momente und akteursbezogener Designentscheidungen: „Wichtig bei diesem Ansatz ist, daß es weniger um formale, vollständige logisch/mathematische Beschreibung und totale Korrektheit geht als darum, daß die konstruierten Systeme ein Verhalten aufweisen, das *vernünftig*, möglichst *optimal* und vor allem *zuverlässig* [...] ist" (Brauer 2001, S. 30).

Wandel des Disziplingefüges der Informatik

Bestandteil etablierter Disziplinen

- Teildisziplin der Mathematik: Technische Mathematik
- Teildisziplin der Nachrichtentechnik:Nachrichtenverarbeitung

Schnittmengenkonstrukte mit mehreren Einflußdisziplinen

- Überlappungsbereich zwischen Mathematik und Nachrichtentechnik ohne Eigengepräge
- Konglomerat aus mehreren Grundwissenschaften der Informatik mit Eigengepräge
- Grenzdisziplin zwischen Mathematik, Technikwissenschaften, Naturwissenschaften,
- Geisteswissenschaften, neuer Wissenschaftstyp

Strukturierter Fächerkanon mit definiertem Wissenschaftskern

- Aussortierung und Ordnung der Teildisziplinen
- Schichten- und Schalenmodelle des Fächerkanons
- Geordnetes, theoretisch begründetes Disziplingefüge: Wissenschaftsgebäude der Informatik, „Informatik-Turm"

Entgrenzung der Disziplin, Verzicht auf tragenden Theoriekern

- Strukturwissenschaft, Systemwissenschaft,
- Querschnittsdisziplin, Grundlagendisziplin
- Konglomerat aus verschiedenen Disziplinen

In diesem Neben- und Ineinander formaler und nicht-formaler Modellbildungen und Methoden treten sich die Teildisziplinen und Theoriebündel nun eher gleichberechtigt gegenüber, die Disziplin wird wieder, wie in der Anfangszeit, zu einem losen Verbund oder Netzwerk. Der algorithmisch-logische Kern hat darin seine Führungsrolle als „Theorie der Informatik" verloren, ohne daß ein Nachfolger in Sicht wäre. Es hat zwar seit längerem schon eine ganze Reihe von Anwärtern für einen nicht-algorithmischen disziplinären Kern gegeben: so die Semiotik, die Wissenstechnik, Interaktions-Modelle, die allgemeine Informationswissenschaft, die Medienwissenschaft und die „allgemeine Modellbildung", die Systemtheorie, die Hermeneutik, die Gestaltungs- bzw. Designwissenschaft, die Organisationswissenschaft und schließlich die Arbeitsgestaltung und Gesellschafts-Informatik. Doch keinem von ihnen ist bisher der Durchbruch als neuem Wissenschaftskern gelungen. Es setzt sich vielmehr, so zeigt der Blick auf den Wissenschaftsdiskurs der letzten Jahre, die Einsicht durch,

- daß sich die Informatik nicht auf einen bestimmten Wissenschaftstyp festlegen läßt,
- daß sie zugleich Grundlagen-, Ingenieur- und Anwendungsdisziplin ist,
- daß sie sowohl formal-logische, empirische, konstruktive als auch hermeneutische und sozialwissenschaftliche Ansätze miteinander verknüpft.

Die Informatik bleibt so, wie Pflüger (1994, S. 251) nahezu in Übereinstimmung mit Zemaneks Einsichten von 1971 feststellt, in ihrem Kern eine Grenzdisziplin, die an den „heterogenen Welten der formalen Modelle und deren sozialer Wirklichkeit“ teilhat.

Das offensichtliche Scheitern der Suche nach einer die Disziplin strukturierenden Kern- oder Leitwissenschaft veranlaßte Peter Denning in den letzten Jahren zum Verzicht auf eine Ordnung im Wissenschaftskonglomerat der Informatik. Anstelle eines wohlstrukturierten Systems von Kern-, Schalen- und Bindestrich-Informatiken setzt er auf eine Entgrenzung des Faches und ein Aufgehen in einer allgemeinen, verschiedene Disziplinen, Gewerbe und Gewerke umfassenden „IT-Profession“. Aus dem einstigen Universalanspruch der Computer Science ist in diesem entkernten Konglomerat „Information Technology“ nur ein bescheidener Bereich theoretisch-methodischer Dienstleistungen für innovative Entwicklungs-, Konstruktions- und Nutzungsprozesse übriggeblieben. Doch gerade durch die Annäherung an Designer und Nutzer und ihre Anwendungswelten, durch den Verzicht auf eine algorithmische Ordnung der Welt nach dem Muster der „Mathesis Universalis“ und die Aufgabe des Führungsanspruchs einer Leitwissenschaft setzt die Informatik, so scheint es, die Diversifikation in andere Wissenschaftsdisziplinen und die Ausdehnung ihrer Wirkungsmacht um so ungebrochener fort. Sie ist am Ende noch viel interdisziplinärer geworden, als es Aiken in seiner Vision der „new discipline“ 1955 vor Augen hatte: Die Informatik ist ebenso „Geistes- wie Naturwissenschaft und Technik. Sie ist ein Brückensystem zwischen den Wissenschaften, denn Sprache und Information sind überall die Grundlage der Arbeit, und die Informatik dient als Werkzeug und Speicher, als Dokumentenquelle und –senke, und sie gehört allen Wissenschaften an, in verschiedenem Maß vielleicht, aber überall beteiligt. (Zemanek 1992, S. 24)

Die Grenzen szientistischer Begründungmuster in der Disziplin Informatik bilden den Hintergrund für die folgende Untersuchung, die die Ergebnisse des technikgenetischen Theorieprojektes „Die Geschichte der Modellierung von Designraum, Designkonflikten und Technologielebenszyklus in der Computer-

architektur“ darstellt. Im Mittelpunkt stand dabei die Wissenschaftsgenese der informatischen Teildisziplin Computerarchitektur. Obwohl heute weitgehend als abstraktes Strukturmodell verstanden, läßt sich die historische Betrachtung informatischer Architekturen einen charakteristischen Konzeptwandel erkennen. Bereits in der Pioniergeneration gab es eine Vielfalt von Modellvorstellungen, die sich entweder an etablierten Technikwissenschaften, an Naturwissenschaften oder an sozialen Organisations-modellen orientierten. Der Modellpluralismus der Frühzeit wurde in den fünfziger Jahren durch Bestrebungen abgelöst, über Anleihen an klassische Ingenieurdisziplinen und das System Engineering aus dem Computer-Design eine ‘normale‘ Technikwissenschaft zu machen. Da die am industriellen Fertigungsprozeß angelehnten systemtechnischen Dekompositionsstrategien des „Computer Engineering“ die Designkomplexität nicht bewältigten, entstand um 1960 im Kreis der Chefdesigner der IBM nun auch unter dem Architekturbegriff das designorientierte Architekturkonzept. Mit der seit Mitte der sechziger Jahre einsetzenden Professionalisierung der „computer architects“ löste sich das Architekturkonzept langsam von der aristokratischen Designkultur und begann mit einer systematischen Erfassung und Strukturierung des Designraums und legte so die Grundlage für die Bauformenlehren, Designkonfliktmodelle und Methodeninventare der sich um 1970 etablierenden Erfahrungswissenschaft Computer Architecture. Ein Ausblick auf die achtziger Jahre zeigt dann, wie durch die Akademisierung streng wissenschaftliche Taxonomien der Bauformen als Grundlage eines theoriebasierten kontrollierten Entwurfs anvisiert wurden. Angesichts der offensichtlichen Grenzen des szientistischen Paradigmas wurden die exakten Dekompositionslehren in den neunziger Jahren wieder von den pragmatischen Engineeringansätzen und einer Renaissance der designbetonten Architekturkonzepte zurückgedrängt. Die Computerarchitektur erweist sich so als konstruktive Technikdisziplin, deren wandelnde Akteurs- und Nutzungskontexte ständige Verschiebungen und Neuaushandlungen der Designmerkmale bewirken, so daß eine an naturwissenschaftsnahen Technikwissenschaften ausgerichtetes Verwissenschaftlichungsmodell hier nur begrenzte Geltung hat.

Die Studie belegt auf diese Weise , daß sich angesichts der Fortdauer der technischen Umwälzungen und der stets noch wachsenden Nutzungs- bzw. Umgebungs-bedingten Komplexität szientistische Methoden der Erfahrungskompression und Designrationalisierung nur begrenzt bewährt haben und daß sich die Disziplin damit wieder mehr als eine konstruktive Technikwissenschaft begreift. Deren Aufgabe besteht wesentlich in Allokationsentscheidungen bei Funktionen und knappen Betriebsmitteln, in der Organisation arbeitsteiliger

Strukturen und Prozesse innerhalb der technischen Aggregate wie im Gesamtsystem sowie generell im Management der divergierenden Nutzer- bzw. Stakeholder-Anforderungen in der Systemgestaltung. Und dieses Multiakteursspiel von Funktionsfindung, Ressourcenallokation und Systemdimensionierung ergibt sich nicht aus noch so systematischen Bauformtaxonomien. Eine streng wissenschaftliche Modellierung nach Vorbild der Mathematik und Naturwissenschaften ist so nicht möglich, die Entwickler müssen hier wie in anderen designintensiven Ingenieurdisziplinen mit einer Vielfalt von Modellsichten und der Heterogenität von Designkonflikten zurechtkommen. Architekturdesign bleibt so ein bewußter heterogener Gestaltungsakt, in dem 'rationale' Strukturbildung, hermeneutische Leitbilddiskurse und Verständigungsprozesse sowie vor allem soziale Aushandlungsprozesse ineinandergreifen. Das Projekt hat so die soziale Genese von Techniken und Technikdisziplinen bis in die inner-technische Modell- und Strukturbildung hinein verfolgt und damit der Wissenschafts- und Technikgeschichte neue
Möglichkeiten eröffnet.

1. Das informatische Architekturkonzept als Forschungsproblem

Grundlegende Systemstrukturen, Bauformen und Stilbildungen sowie Entwurfsmethoden und Designreflexionen werden in der Computer Community seit ca. 1960 mit der Architektur-Metapher belegt, zunächst ausschließlich im Bereich der späteren Rechnerarchitektur. Entstanden im Kontext von Professionalisierungsbestrebungen der Chefdesigner, entwickelte sich der Begriff "computer architecture" bald zur umfassenden Kompositions- und Bauformenlehre, die sich im Laufe der 70er Jahre als eine Teildisziplin der Computer Science etablierte.[2] Im Zusammenhang mit Problemen der Bewältigung der Komplexität von Softwaresystemen, insbesondere von großen Betriebssystemen, erhielt der Architekturbegriff dann ab 1965 auch Einzug in die Softwarekonstruktion, hier von Anfang an mit deutlicher Tendenz zur hierarchischen Strukturierung, Modularisierung und Standardisierung. Doch gerade wegen der Zielvorstellung einer Wissenschafts-basierten Dekompositionslehre konnte sich "Softwarearchitektur" gegenüber dem seit 1970 vorherrschenden Leitbegriff "Software Engineering" nicht durchsetzen. Erst seit dem letzten Jahrzehnt beginnt er sich als ein stärker gestaltungsorientiertes Gegenkonzept zum Produktionslinien-Modell der Software-Entwicklung einzubürgern. Über verteilte Rechnerarchitekturen und Strukturierungsansätze bei der Netzwerksoftware wurde der Architekturbegriff 1969/70 auch von der Computer Network Community adaptiert, zunächst allgemeiner orientiert auf alle Bereiche der Netzwerkgestaltung, seit Mitte der 70er Jahre aber mit klarem Übergewicht der hierarchischen Strukturierung und Modulbildung. Auch die Netzwerkarchitektur wurde vorwiegend zu einer Dekompositionslehre, die besonders der Entwurfsrationalisierung und noch mehr der Gewährleistung von Kompatibilität dienen sollte. Seit den Siebzigern etablierte sich der Architekturbegriff in der Computer Science endgültig, er wurde zu einer von der gesamten Fachwissenschaft anerkannten Bezeichnung für Systemstrukturen und Teildisziplinen, wobei er sich in den Bereichen Computerarchitektur und Netzwerkarchitektur am nachhaltigsten durchsetzte. Seit den 80er und 90er Jahren erfolgte dann eine Ausdehnung des Begriffes auf die gesamte Informatik und Informationstechnik, und neuerdings auch auf die allgemeine Ingenieurwissenschaften (Broy, Schmidt, 1999, S. 206). Architektur wurde nun zu einem Universalbegriff für jede übergeordnete Strukturierung, Stilbildung und Gestaltungsphilosophie.

---

[2] Zum Begriff der Kompositionslehre siehe Kesselring 1954.

Mit dieser Anlehnung an eine künstlerisch-gestalterische Profession stehen Informationstechnik und Informatik ziemlich einzigartig in der ingenieurwissenschaftlichen Theorie- und Methodenlandschaft da, denn außerhalb der eigentlichen Architektur und Bautechnik hat der Begriff sonst über gelegentlichen metaphorischen Gebrauch hinaus kaum Fuß fassen können. Trotz dieser Sonderstellung und der nun schon jahrzehntelangen Verbreitung existieren bislang fast nur bereichsinterne und rein gegenstandsbezogene Aufarbeitungen des Wandels von "Architekturen" (vgl. bes. Smith 1989; Ceruzzi 1998), aber kaum übergreifende vergleichende Analysen und Reflexionen über die sich in Architekturbegriffen vollziehenden impliziten Theoriedebatten und Paradigmenwechsel. Dies ist erstaunlich, wenn man diese Forschungslücke mit den ausgiebigen theoretischen Erörterungen über den Gestaltungsbegriff in der Informatik oder über das "Engineering-Konzept" in der Softwarekonstruktion vergleicht. Engineering- und Architektur-Metapher zielen beide gleichermaßen auf die Bewältigung der Vielgestaltigkeit von Lösungsräumen hard- und softwaretechnischer Systeme und auf die damit zusammenhängende Komplexität von Designentscheidungen. Doch verdient es genauerer Untersuchung, warum man auf der einen Seite Konzepte zur Ordnung, Strukturierung und Organisation von Designräumen mit eher analytisch ausgerichteten ingenieurwissenschaftlichen Begrifflichkeiten belegt und auf der anderen eher ganzheitlich-synthetisierende Disziplinen als Vorbild wählt. Letzlich geht es dabei auch um die Verortung wichtiger Teildisziplinen der Informatik im System der Wissenschaften.

Der folgende Beitrag rekonstruiert die Genese der konzeptionellen Auffassungen in der Computer Community über Rechnerdesign, Rechnerstrukturen und -bauformen, die seit den 60er Jahren mit dem Architekturbegriff belegt wurden. Dabei geht es ausdrücklich nicht um den historischen Wandel der Rechnerarchitekturen selber, sondern um eine Metabetrachtung der Entwicklung von Architektur-Auffassungen in den Fachdiskursen. Gegenstand ist der implizite Konzeptwechsel, die meist stillschweigende Verschiebung der Modellintentionen und Methodikziele, insbesondere der jeweilige Wandel von einer eher künstlerisch-ganzheitlichen Designlehre zu einer vorwiegend auf Entwurfsrationalisierung und Produkt- bzw. Prozeßstandardisierung zielenden Strukturlehre und Bauformensystematik, aber auch auf immer wiederkehrende Bestrebungen, dem Trend zur Verwissenschaftlichung und Algorithmisierung eine Design-betonte Architekturauffassung entgegenzusetzen. Der Untersuchungszeitraum reicht dabei vom Beginn der modernen Computerentwicklung in den 30er Jahren bis zur Etablierung der Fachdisziplin nach 1970 mit einem Aus-

blick bis zur Gegenwart. Als Quellengrundlage dient die umfangreiche Literatur der *Architekturtraktate* als herausragende Form der "Literalisierung" der "architectural community". Deren Spannweite reicht von grundlegenden Designreflexionen und Erfah-rungsberichten der Computerentwickler über Bauformen-Vergleiche und State-of-the-Art Reports auf Fachtagungen bis zu elaborierten Lehr- und Fachbüchern. Untersuchungsmethode ist die historisch-vergleichende Textinterpretation, die in ähnlicher Weise bereits in Studien zur Geschichte der Konstruktionsmethodik im Mechanik- und Elektrobereich erprobt wurde. (Hellige 1991, 1995 a/b) Auch bei der Analyse der Architekturkonzepte soll es um die Frage gehen, nach welchen wissenschaftlichen Vorbildern die neue Technikdisziplin modelliert wird, welche sozialen Organisationsleitbilder in die Produkt- und Prozeßmodelle eingehen und wie die Probleme der Komplexität des Lösungsraumes und der Designkonflikte gelöst werden sollen.[3]

Den theoretischen Hintergrund der Analyse bilden Modelle der Entwicklungslogik und -stufen der Technikwissenschaften in der Wissenschaftssoziologie und -geschichte. Die entsprechenden empirischen Analysen haben sich vorzugsweise mit den naturwissenschaftlich orientierten Ingenieurdisziplinen beschäftigt, bei denen stoff- und energieumwandelnde oder verfahrenstechnische Prozesse im Mittelpunkt stehen und die daher weitgehend dem Wissenschaftsverständnis der Physik und Mathematik folgen.[4] Das Resultat sind hier meist an Kuhn angelehnte lineare Stufenkonzepte der Theoretisierung und Verwissenschaftlichung, die von Kunstlehren und der Erfahrungsakkumulation über die Entwicklung heuristischer und empirischer Methoden zu umfassenden Taxonomien und theoriebasierten Wissenschaftssystemen reichen. Demgegenüber stehen die mehr konstruktiven Ingenieurdiziplinen mit stärkerer Affinität zu Designlehren und Konstruktionsmethodiken eher am Rande der wissenschaftssoziologischen Betrachtung.[5] Zwar existieren auch dafür lineare Entwicklungsmodelle vom kunst- und handwerksartigen Entwurf bis zum Design-Algorithmus bzw. zu Konstruktionsverfahren auf streng wissenschaftlicher Grundlage nach dem Muster der Chemie (Oelsner 1992 a/b), doch hat

---

[3] Diese Studie wurde angeregt durch das unerwartete Interesse von Informatikern an meinen Untersuchungen zur Konstruktionsmethodik des Mechanik- und Elektrobereichs, er wurde gefördert durch viele intensive Diskussionen mit Hartmut Petzold, Hans-Ulrich Niemitz, Kpatcha Bayarou und vor allem Jörg Pflüger, der die verschiedenen Fassungen mit kritischem Rat begleitet hat. Last not least bekam ich wertvolle Hinweise durch eine ganze Reihe von Gesprächen mit Heinz Zemanek, N. Joachim Lehmann und Ambros Speiser.

[4] Vgl. hierzu u.a. v. d. Daele, Krohn 1978; Banse, Wendt 1986; Buchheim 1980; König 1987, 1993

[5] Zu den Typen von Technikwissenschaften siehe Sonnemann 1987; Banse, Wendt 1986, S. 15-18

sich das Verwissenschaftlichungs-Paradigma hier nie vollständig durchsetzen können. So findet man in diesem Bereich auch komplexere Entwicklungsmuster, die von Rückbewegungen zu Gestaltungskonzepten oder Pendelbewegungen zwischen verstärkter Wissenschafts- und Designorientierung ausgehen (vgl. hierzu Hellige 1995b, König 1999).

Bedeutungen des Architekturbegriffes im Bereich der Rechnerarchitektur

- **Architektur als Kernstruktur der Hauptkomponenten**
- **Architektur als Gesamtstruktur des Computersystems**
- **Architektur als oberste Ebene des Rechnersystems**
- **Architektur als Residualgröße (nicht-formalisierbarer Designbereich)**
- **Architektur als Designmethodik und Kunstlehre**
- **Architektur als individueller Stil eines Computerdesigners**
- **Architektur als individueller Stil eines Rechners**
- **Architektur als Stil einer Rechnerfamilie**
- **Architektur als Schnittstelle zum Benutzer**
- **Architektur als Design- und Benutzungsqualität**
- **Architektur als Blaupause, Designleitfaden für den Rechnerentwurf**
- **Architektur als Gesamtheit von Hardware- und Softwaresystem (einschließlich Rechnerorganisation)**
- **Architektur als charakteristische Bauform, Merkmalkombination, Konfigurationstyp**
- **Architektur als jeweilige Schnittstelle zwischen den Ebenen**
- **Architektur als hierarchisches Schichten-/Ebenenmodell des Gesamtsystems**
- **Architektur als wissenschaftliches System der Taxonomie/ Klassifikation von Rechnerbauformen und -bauweisen**

Auch innerhalb der Informatik stehen sich szientistische und designorientierte Entwicklungsmodelle gegenüber. So macht etwa für Karl Ganzhorn (1974, S. 42) die Lehre von der "Rechnerstruktur" und die Computer Science insgesamt den typischen Entwicklungsprozeß von Wissenschaftsdisziplinen durch, nämlich vom Zusammentragen der empirischen Fakten zu einem Wissenskomplex über die Beschreibung individueller Maschinen und generalisierende Abstraktionen zur ausgereiften Wissenschaft, in der das gesamte Wissen "in

eine endliche Zahl von Grundprinzipien oder invarianten Grundfunktionen zurückführbar gestaltet ist." Um zu einer "strukturierten Ingenieurwissenschaft" zu werden, sollten die "Prinzipien der Rechnerstrukturen" vor allem auf die strengen Theorieansätze der theoretischen Informatik zurückgreifen, insbesondere auf die Informationstheorie, Automatentheorie, Graphentheorie und die exakten formalen Sprachen. Erst durch solche "invariante Säulen" - d.h. "eher praxiserprobte Konstruk-tions-prinzipien als dem Dunkel entrissene Gesetze" - bilde sich eine stabile Basis- und Ingenieurwissenschaft heraus, die den einzelnen Entwicklungen nicht mehr nachlaufen muß (Ganzhorn, Schulz, Walter 1981, S. 6 f.). Ganz ähnlich sieht Robert R. Johnson in einer Verknüpfung von Informations- und Entscheidungstheorie und Petrinetzen die geeignete Grundlage einer Theoretisierung des Computerdesigns. Durch die Vereinheitlichung der Repräsentation von Rechner- und Programmstrukturen, Problembeschreibung und Problemlösung werde die Rechnerarchitektur Teil eines übergreifenden Systemdesigns, dem Reifstadium der Informationsverarbeitung (Johnson, 1974, S. 3 ff.).

Demgegenüber vertritt Heinz Zemanek (1980, 1986) ein betont designtheoretisches Modell der informatischen Architekturentwicklung, das sich im Kern als ein Phasenkonzept grundlegender Designtypen darstellt. Hierbei folgt auf die "natural phase", "builder's phase" und "architecture phase" noch die Phase der "Re-Humanisierung der abstrakten Architekturen", in der Nutzer- und Gesellschafts-bezogene Designfragen wieder einen zentralen Stellenwert erhalten. Der Rekurs auf die Architekturtraktate soll im folgenden darüber Aufschluß geben, welche der Modellvorstellungen der Disziplinentwicklung der Rechnerarchitektur adäquater ist. Er soll auch einen neuen Zugang zur Vielgestaltigkeit der Entwicklungsformen von Technikwissenschaften eröffnen, die durch die Modellfixierung und die in der Wissenschaftssoziologie bevorzugte Sekundärauswertung von Darstellungen aus dem Blick geraten ist.[6]

[6] Dies zeigt sich besonders bei Hohn 1998.

Vorempirische Architekturkonzepte der Pionierphase

Entwicklung der grundlegenden Organstruktur: Zuse 1936-44

- Definition der Aufgabenstruktur der "Organe der Rechen-maschine" und der Designschritte
- Unterscheidung der logischen "Verfahren", der "Grundform des Aggregats" und der "konstruktiven Durchbildung der Maschine"

Designkonzept nach Vorbild der Elektrokonstruktion: Aiken 1937

- Logische Trennung des "whole problem of design" von der Anlagen- und Organstrukturierung und der Ausführung mit unterschiedlichen Bauelemente-Technologien
- Rudimentäres Phasenmodell der Computerkonstruktion

Elementare Organstrukturlehre und axiomatische Designmethodik: v. Neumann, Burks, Goldstine, 1945-48

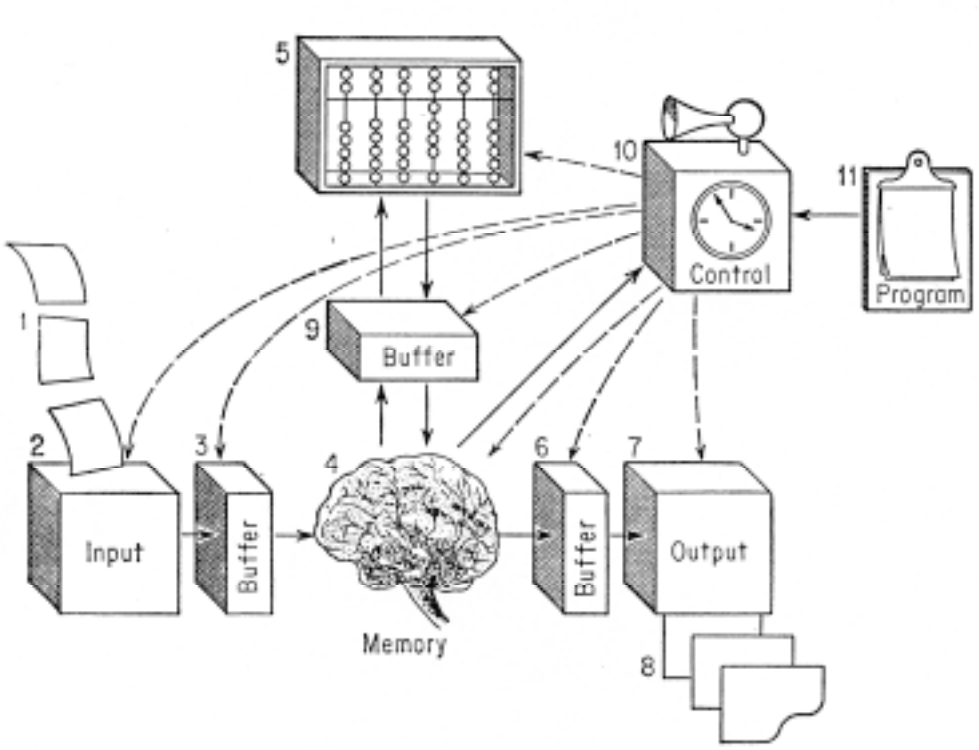
- Entwicklung eines "complete logical framework for the machine" und der "design structure" eines Einprozessor-Rechners als elementare Systemkonfiguration
- Minimale Organstruktur auf logischer Ebene als "One-best-way-Lösung"
- Rechnerkonfiguration auf der Designebene als komplexe Problem und Zielkonfliktstruktur infolge der Interdependenz der Systemkomponenten und der Vielfalt der Lösungsvarianten bei der "organization" der Kernorgane
- Trade-off-Charakter des Rechnerdesigns und Kompromißcharakter der Designziele Geschwindigkeit, Verfügbarkeit, technischer Aufwand und Kosten
- Das "system of organs" als systemisches Gleichgewichtsmodell und als Basis der Dimensionierungs-Entscheidungen ("balance of the machine")
- Architekturartige Bezeichnungen für die übergeordneten Strukturmerkmale und Designaspekte: "logical Structure" , "logical design", "design structure", "structure of a computing system", "organization of a computer"

Logisches Universalmodell des Rechners und nutzungsbezogenes Designkonzept: Alan Turing, 1936/46/47

- Denkmodell für universale Informationsverarbeitungs-Maschinen
- Integration von elementarer Kernstruktur und rechner-spezifischem Designphasenmodell

Schichtenmodell des Computers u. Ansätze für ein Life-Cycle-orientiertes Organisationskonzept des Designs : George Stibitz, 1947/48

- Ausweitung des Designkonflikt-Ansatzes auf die Nutzungs- und Endphase
- Multiperspektivische Sicht des Rechnerdesigns: interne und externe Organisationsaufgaben beim Entwurf
- Das Computersystem als Hierarchie kooperierender Schichten



Organ- / Organisationsmodell der ‚von-Neumann-Architektur' (Flores 1960, S. 41)

2. Vorempirische Architekturkonzepte der Pionierphase: Der Computer als elementare Organstruktur und Organisationsproblem

Mit die frühesten theoretischen Erörterungen im Rahmen konkreter Rechnerentwicklungen finden sich bei Konrad Zuse. Bereits 1936 widmete er sich in Vorarbeiten oder Entwürfen für Patentanträge grundlegenderen Fragen über das Zusammenspiel des von ihm von vornherein in Aussicht genommenen "dyadischen" Zahlensystems mit den Hauptkomponenten seiner programmgesteuerten, freiprogrammierbaren Rechenmaschine: der ″Rechenvorrichtung″, des ″Speicherwerks″ und des ″Wählwerkes″, das die Verbindung des Rechenwerkes zu bestimmten Speicherzellen herstellte, sowie des "Rechenplans", d.h. der ″Vorrichtungen zum Steuern der Anlage durch Lochstreifen″ (Zuse 1936a). In der Patentanmeldung von 1936 und noch mehr in der von 1941 betrachtete Zuse den ″konstruktiven Aufbau″ der ″Gesamtanlage″ bzw. die "Gesamtanordnung der Vorrichtung" sowie Art und Aufgaben der ″Organe der Rechenmaschine″ bereits aus einer architektonischen Perspektive. Er arbeitete die Strukturabweichungen gegenüber traditionellen Rechenmaschinen deutlich her-

aus und unterschied dabei auch schon klar zwischen dem "Grundprinzip" bzw. der "Grundform des Aggregats" und der "praktisch durchgeführten Lösung", d. h. der "konstruktiven Durchbildung der Maschine", bei der viele "Abwandlungen" möglich sind (Zuse 1936b, 1941; Petzold 1998, bes. S. 74). Mit der Separierung der ″Organe Rechenwerk, Speicherwerk und Leitwerk″ und der Darlegung des Zusammenhangs von apparativem Ablauf in Gestalt eines Flußdiagramms und der Durchführung des 'Programmiermodells' des ″Rechenplans″, den er bereits vom Prinzip her für speicherbar hielt, erfüllte Zuse schon ein halbes Jahrzehnt vor dem EDVAC-Team einen Großteil der Kriterien der ″von-Neumann-Architektur″, auch wenn es sich bei seinen konkreten Entwürfen im strengen Sinne noch nicht um Universalmaschinen handelte (Bauer 1996; Rojas 1998, S. 57 f.).

Durch das Dualsystem gelangte Zuse sehr bald zu abstrakt-theoretischen Fraugen der mathematischen Logik und erhielt dadurch - trotz des hartnäckigen Festhaltens am Begriff der "Rechenmaschine" - ein viel umfassenderes Verständnis von "Rechnen": es bedeutete für ihn die Bildung neuer Angaben aus vorhandenen mittels einer Vorschrift, wobei "Angaben" Zahlen, Aussagen, Namen, Kennziffern, Befehle, Nachrichten usw. sein konnten.[7] Tagebuchnotizen, die er in seinen Memoiren zitierte oder die jetzt im "Zuse-Internet-Archiv" zugänglich geworden sind, legen den Schluß nahe, daß mit der Vision eines "mechanischen Gehirns", der Unterscheidung von "starren und lebenden Rechenplänen" sowie von "äußeren und inneren Befehlen" das "Stored Program-Konzept" bei Zuse schon zwischen 1937 und 1939 voll ausgebildet war.[8] Doch ist die genaue Datierung dieser Dokumente nicht in allen Fällen zweifelsfrei verbürgt. Bei manchen dieser oft nachträglich datierten Aufzeichnungen[9] handelt es sich , wie die rückschauende Perspektive der Eintragungen und die offensichtliche Verwendung von Begrifflichkeiten späterer Jahre in den Transskriptionen (z.B. "universelle Programmiersprache", "allgemeine Formelsprachen" und "bedingter Sprung") belegen, um nachträgliche Rekonstruktio-

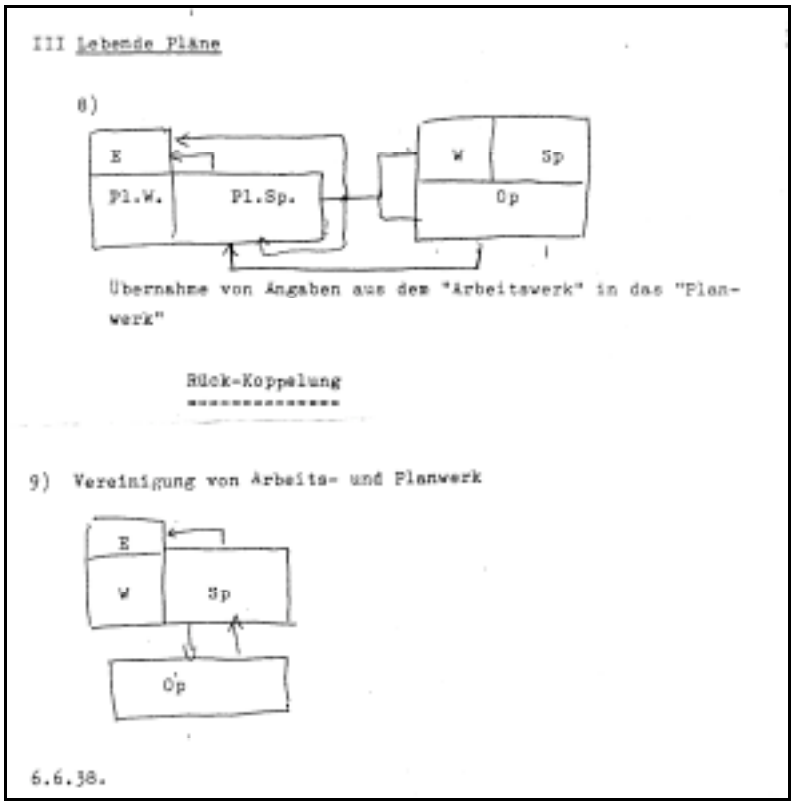
---

[7] Zuse 1938a, 1943; dieser Zusammenhang wird besonders prägnant bei Petzold, 1985, S. 321-326 herausgearbeitet.

[8] Siehe Zuse, Tagebuch, 20.6.1937 in: ders., 1984, S. 40 f.; Zuse, Anlage zur Tagebuchnotiz vom 4.6.1938 über Programmspeicherung, Zuse-Internet-Archiv, 0311; Zuse, Tagebuchnotiz vom 25.5.1939 über flexible Aufgabenstrukturen, Zuse-Internet-Archiv, 0418

[9] Vgl. D. Dotterweich, Zuse-Dokumentation, S. 92: Für die Mikroverfilmung einer Auswahl von 600 Dokumenten aus den Nachlaßpapieren, die die Basis der GMD- bzw. HNF-Dokumentation bildet, wurden ab 1978 die Unterlagen gesichtet, geordnet und großenteils datiert: "Es fehlten zum Beispiel sehr häufig die Datumsangabe beziehungsweise das Entstehungsjahr, Herkunftsangaben und ähnliches. Da sehr viele Dokumente nur als Stenogramm vorhanden sind, war auch eine Inhaltsanalyse für diese Dokumente von ihm erforderlich".

nen, spätere Inhaltsangaben mit Begriffsaktualisierungen oder, wie im Falle der Plankalkül-Publikation von 1972, um Überarbeitungen.



Organe der Z3 mit Rückkopplung von „Arbeit"- und „Planwerk"
Transkription von Eintragungen vom 5./6.6.1938 in Zuses Notizbuch (Zuse 1938b)

Die Universalität seines Rechnerkonzeptes lag so allem Anschein nach vor 1945 nicht in einem universellen Computer mit intern gespeicherten und modifizierbaren Programmen, sondern in der Vorwegnahme eines Anwendungsspektrums, das in mancher Hinsicht die Computernutzung der fünfziger und sechziger Jahre vorwegnahm. Für Zuse war nämlich seit circa 1937/38 die „Rechenmaschine"– im Kreis der Freunde und Mitarbeiter sprach er selber von einer „Universal-Rechenmaschine" (Grohmann 1979, S. 88) - nicht mehr nur ein zahlenverarbeitender Apparat, sondern eine allgemeine ‚algorithmische Maschine', die „Angaben" oder „Umstände" nach „Vorschriften" veränderte. Es konnte sich dabei um Zahlen, Buchstaben, sonstige Zeichen oder Symbole handeln, die für Kontrollinformationen, Arbeitsprogramme, Sortierkategorien, Warenpositionen oder Personenmerkmale standen.[10]. Die Einbindung in die

[10] ZuB044/006

Kriegswirtschaft des Dritten Reiches bewog Zuse, die Bestandsführungs- und Kontrolltechniken der Lochkartenwelt mit dem technisch-wissenschaftlichen –Berechnungsmöglichkeiten des freiprogrammierbaren Computers zu verknüpfen. Daraus entstanden bei ihm die umfassenden Konzepte der „automatischen Behandlung schematisch-kombinatorischer Aufgaben“ und der Erfassung des „ganzen schematischen Wissens“ in ständig aktualisierbaren „Speicherwerken“. Zuses Perspektive war hierbei nicht mehr die eines Rechenmaschinen-Erfinders, der nach neuen Anwendungen Ausschau hielt, sondern bereits die eines ‚Informatikers‘, der auf der Basis des Logikkalküls zu einer „allgemeinen Theorie des Rechnens“ vordringt.[11]

Durch den theoretischen Zugang erschlossen sich ihm sehr früh Einsatzfelder, die den auf „Number Crunching“ fixierten amerikanischen Erfindern verborgen blieben. Bereits während der Z1-Entwicklung erörterte er im Kreis seiner Helfer diverse Anwendungsmöglichkeiten: „Aber Zuse machte uns klar, daß Rechnen nur ein Spezialfall logischer Operationen ist und daß sein Apparat z.B. auch Schach spielen können müsse.“ (Buttmann 1979, S. 87) Weitere Ideen waren die Verdrahtung der Rechner, die Wetterberechnung, Währungsrechnung und stets aktuelle Kontenführung und Tagesbilanzierung bei Banken, die Mechanisierung des Warenhausbetriebes, ja ganzer Warenwirtschaftssysteme, die Personalabrechnung und „Gefolgschaftskontrolle“, die Überwachung und Steuerung von Maschinen und Materialwirtschaft, die Bahnsteuerung von Werkzeugmaschinen wie die rechnerunterstützte Maßschneiderei und bald auch die Zeichenmaschine. Durch die „ständige Bereithaltung von Angaben in Speicherwerken“ sollte in begrenztem Umfang Ingenieur- und Konstrukteurwissen wie auch Arbeitsprogramme der Fertigung abrufbar und wieder verwendbar werden, wobei er sich offen-sichtlich Illusionen hinsichtlich des erforderlichen Speicherplatzes machte. Aus der Bekanntschaft mit den wehrwirtschaftlichen Lochkarten-Fernschreibprogrammen entstanden bei Zuse - entgegen der von ihm später immer wieder geäußerten Skepsis gegenüber Rechnervernetzung - bereits Visionen und Konzepte für eine betrieblich, lokal oder national vernetzte Informationsverarbeitung. Der Rechner war für Zuse so schon Anfang der vierziger Jahre zugleich ein Rechenautomat, ein technisch logistisches Programmierungs- und Steuerungsinstrument, ein Kontrollmedium und schließlich ein Datenkommunikationsmedium.

---

[11] Siehe hierzu und zum folgenden bereits Petzold 1985

Trotz der dringend erforderlichen quellenkritischen Klärung der Zeitschichten anhand der Originalnachlaßpapiere bleibt es unbestritten, daß Zuse schon in den Jahren 1939-44, d.h. vor der Von-Neumann-Gruppe, die logische Abstraktion "des allgemeinen Rechnens" im Blick hatte und nicht, wie die meisten anderen frühen Erfinder, hauptsächlich die maschinelle Apparatur und die Konstruktionstradition der Elektrotechnik. Doch da ihm zum damaligen Zeitpunkt weder Babbage noch andere Vorläufer programmierbarer Rechenautomaten wie auch kaum vergleichbare Entwicklungen in der Gegenwart bekannt waren, gelangte er mangels Vergleichsmöglichkeiten nicht wirklich zu konzeptionellen Architekturreflexionen. Erst nachdem er von den "amerikanischen Großgeräten" erfahren hatte, fühlte er sich im Oktober 1947 veranlaßt, öffentlich auf die "besondere Theorie der Zuse-geräte" und seinen "mathematischen Formalismus" hinzuweisen, der "es gestattet, allgemeine kombinatorische Denkaufgaben - frei von jeder Spezialisierung auf ein bestimmtes Aufgabengebiet - in eine allgemeine mathematische Form zu bringen, welche den Ausgangspunkt für die maschinelle Lösung derartiger Probleme darstellt" (Zuse-Ingenieurbüro Hopferau, 1947, Teil IV). Doch da Zuse nicht mehr dazu kam, die begonnene Dissertation abzuschließen und damit seinen architektonischen Ansatz im Zusammenhang zu entwickeln, erlangte er niemals eine den Traktaten von Burks, Goldstine und von Neumann vergleichbare systematische Darstellungsform.

Die bei den Computerpionieren sonst übliche, eher traditionelle Sichtweise zeigen die konzeptionellen Erörterungen Howard H. Aikens in seinem ″Prospectus″ aus dem Jahre 1937, der den Ausgangspunkt seines 1939-44 gebauten Relais-Rechners Mark I bildet und der als einziger Text von ihm aus der Frühzeit überliefert ist (Aiken 1937; Cohen 1999, S. 37 f.). Nach Vorbildern der designmethodisch bereits viel weiter vorangeschrittenen Elektrokonstruktion entwickelte der ehemalige Elektroingenieur Aiken das an Babbages "Analytical Engine" angelehnte Konzept einer ″Automatic Calculating Machine″ als eine von der Hardware-Realisierung unabhängige logische Struktur. Bei der Schilderung der erforderlichen Entwicklungsschritte legte er bereits implizit ein rudimentäres Phasenmodell zugrunde: Bedarfsanalyse und Ermittlung der ″requirements″ für den gedachten Einsatzzweck, Bestimmung der Funktionalität unabhängig von der technischen Ausführung, Identifikation des technischen Kernproblems und des zentralen Lösungsansatzes für den ″switching mechanism″ auf Basis der ″Switchboard-Metapher″ und schließlich die Auflistung der Teilkomponenten und Skizzierung des für ihn sehr wichtigen Arbeits- und Bedienkonzeptes. Die eher konservative Designmethodik ging

dabei mit einem dezidiert frühen und starrem "freezing" des Entwurfs einher. Aiken differenzierte das Phasenmodell des Konstruktionsprozesses später weiter aus und legte es auch seinen Kursen zur 'Computer Science' in der Harvard-Universität zugrunde, wobei die Unterscheidung des "functional" und "operational point of view" von der Bauelementesicht des Computers den methodischen Kern bildeten: : "The design of a computing machine is understood to consist in the outlining of its gene-ral specifications and the carrying through of a rational determination of its functions, but does not include the actual engineering design of component units."[12]

Durch die Anlehnung an die Designmethoden und -prinzipien der Elektro-, Elektronik- und Feinwerk-Konstruktion kam Aiken schon Ende der 30er Jahren zu einer Top-down-Sicht des Rechnerdesigns und zu ersten methodischen Grundsätzen: Er arbeitete die strukturellen Unterschiede elektronischer Rechenautomaten zu Hollerith-Systemen heraus und identifizierte das "problem of suitable control design" als Kernproblem des Computer-Entwurfs. Er gelangte zur logischen Trennung des ″whole problem of design″ (Aiken 1937, S. 196) von der Anlagen- und Organstruktur und der Ausführung durch wahlweise mechanische, elektromechanische oder elektrische Baugruppen und entwickelte schließlich ein rudimentäres Designphasen-Modell.[13] Doch blieb Aiken andrerseits gerade durch seine Verankerung in der konstruktionsmethodischen Tradition der Elektrotechnik noch im Vorfeld eigentlicher Architekturbetrachtungen (vgl. Cohen 1999, S. 58 f.).

Dies gilt noch mehr für John Atanasoff, dem Erbauer des ersten Prototypen eines *elektronischen* Computers. Atanasoff lieferte erst 1940 eine Expost-Darstellung über den schon seit 1936 erwogenen und 1937 laufenden Designprozeß für den binären elektronischen Atanasoff-Berry-Computer (ABC) und legte dabei aus der Rückschau auch entscheidende Designschritte, Bauprinzipien und Designkriterien dar. Doch trotz der Formulierung erster ″general principles of computing machine″ und der ″essential elements of computer″ blieb er noch überwiegend auf die eigene Maschine und ihr spezielles Einsatzfeld fixiert, nämlich ″large systems of linear algebraic equatiums″. [14]

---

[12] Zit. nach Cohen, Howard Aiken and the Dawn of the Computer Age, in: Hashagen, Rojas,
[13] Vgl. auch die ausführlicheren mit Grace Hopper verfaßten Darstellungen nach Fertigstellung der Anlage von 1946, wiedergedruckt bei Randell S. 199-218
[14] Atanasoff, John V., Computing Machine for the Solution of Large Systems of Linear Algebraic Equations, (1940), gedruckt in: Randell, Brian (Hrsg.), The Origins of Digital Computers. Selected Papers, Berlin, Heidelberg, New York 1973, S. 305-325; vgl. auch Burks, ENIAC, 1981, Annals, S. 325

Zu grundsätzlicheren Architektur-Reflexionen kam es in den USA erst im Zusammenhang mit Designdiskussionen über den ENIAC und vor allem über dessen Nachfolgemaschine, den EDVAC. Die Riesenanlage des ENIAC selber, die erste Realisierung eines programmierbaren Rechners in elektronischer Technologie im Großmaßstab, war zunächst offenbar kaum Gegenstand grundsätzlicher Reflektion. Deren Chefdesigner John Mauchly und John Presper Eckert beschränkten ihre Designüberlegungen zwischen 1941/42 und 1944/45 im wesentlichen auf Analogien bzw. Abgrenzungen zu den den Systementwürfen zugrundeliegenden mechanischen, elektromechanischen und elektronischen Vorbildern, insbesondere zu Irwin Travis' Ansatz verkoppelter Rechenmaschinen, zum "Differential Analyzer" von Vannevar Bush und zur "Computing Machine" von John V. Atanasoff.[15] Gegenüber diesen Rechnern strichen sie den "general purpose"-Charakter bzw. die Vorteile der Verwendung elektronischer Bauelemente in ihrem "electronic digital analyzer" heraus, nämlich die Gewinne für die Designmerkmale Geschwindigkeit, Genauigkeit, Korrektheit und Testbarkeit sowie für die Flexibilität gegenüber neuen Aufgaben durch Neuverknüpfung der Komponenten über ein "program device".[16]

Darüber hinaus bildete die polymorphe Organstruktur der Dezimalmaschine ENIAC aber offensichtlich keinen großen Anreiz für weitergehende architektonische Vorabreflexionen. Die Rechenanlage beruhte auf dem Konzept einer "group of interconnected submachines", die aus 30-40 großenteils parallel arbeitenden Einheiten bestand, die selbst digitale Analogien zu Komponenten mechanischer Rechenmaschinen und Intergrieranlagen darstellten.[17] Jede "unit" verfügte über eigene "program controls", die über den zentralen "master programmer" koordiniert wurden. Die Anlage wurde nach einem frühen "freezing" des "system design", das vor allem auf Mauchly zurückging, von den Bauelementen her aufwärts über die Hauptkomponenten zur Kontrollstruktur entwickelt. Die beteiligten Ingenieure hatten dabei im Rahmen eines "set of rules" freie Hand.[18] Zusätzliche funktionale Anforderungen der Militärs und entdeckte Unzulänglichkeiten führten zu einem grundlegenden Redesign des ursprün-

---

[15] Vgl. Burks, Burks 1981, bes. S.316 f., 333 ff., 344f.; Marcus, Akera 1996 (Die vielen Hinweise und Anspielungen in verschiedene Richtungen sprechen m. E. gegen eine Festlegung auf eine einzige Metapher.)

[16] Mauchly führte hier 1942 den Programm-Begriff in die Computertechnik ein, vgl. Grier 1996; Eckstein 1996, bes. S. 41f.; Mauchly, K. R. 1984, bes. 134 ff.

[17] Albert A. Bennett, 1943, zit. nach A. W. und A. R. Burks, 1981, S. 334; vgl. Report on the ENIAC, 1946, I,1.

[18] Vgl.. bes. Eckert, 1980, bes. S. 526 ("Our main method for guiding a group effort, involving only a dozen engineers, plus technicians and wireman, was to give a set of rules, a kind of discipline, to work by").

glichen Entwurfes mit mehr als verdreifachtem Hard-ware-aufwand (Lukoff, 1979, S. 33-36) Erst danach wurde das eigentümliche Programmiermodell ausgearbeitet, bei dem jeweils für neue Berechnungen unzählige Daten- und Kontrolleitungen ("program cables") rekonfiguriert und verbindungsprogrammiert werden mußten. Mit seiner offenen Struktur und der bestimmenden "idea of parallelism" war dieser flexible Verbund kommunizierender Recheneinheiten als Ausgangspunkt für ein Architekturkonzept daher denkbar ungeeignet (H. H. und A. Goldstine, 1946, bes. S. 15).

Erst nach Abschluß der Bottom-up-Entwicklung und bereits im Zusammenhang mit den Überlegungen für ein Nachfolgemodell stellten die ENIAC-Designer systematischere Überlegungen über Struktur, Operations- und Bedienkonzept der Rechenanlage an. Vor allem Eckert gelangte im Rahmen der Moore-School-Lectures 1946 aufgrund von Verglei-chen mit dem EDVAC und den Maschinen von Aiken und Stibitz zu einer architektonischen Sicht der ENIAC-"Philosophy": Der Designprozeß erscheint nun im Nachhinein als eine Top-down-Entwicklung von der "idea" und dem "plan", über die Festlegung der "logical structure", der "logical basis of the design", zur "materialization of these ideas" in der "physical structure". Eckert beschrieb die Grundstruktur des ENIAC jetzt als "parallel machine" und unterschied sie von den Bauweisen der "serial machines" und von "truly parallel machines". Er verglich die Vor- und Nachteile der Bauformen im Hinblick auf Geschwindigkeit, Bauelementeaufwand und Kosten sowie Konfigurier- und Programmieraufwand und leitete daraus "general statements", d. h. Designprinzipien ab.[19] Grundlage seiner Struktur- und Designbetrachtungen und der Dimensionierungsempfehlungen war ein Gleichgewichtsmodell der unterschiedlichen Prozeßabläufe von Rechen- und Bedienoperationen, aus dem sowohl die "balance of speed" als auch ein "overbalance of equipment" abgelesen werden kann. Aus der Perspektive der Proportionalität der "principle parts of the machine" und der Ausgewogenheit der Dimensionierung erschien nun auch Eckert der ENIAC als ein problematisches Design, das er mit dem letztlich doch recht begrenzten Verwendungszweck und der kriegsbedingt überstürzten Entwicklung begründete (Eckert 1946a, S. 117).

Die Fortschritte im architektonischen Reflexionsniveau bei den ENIAC-Entwicklern waren indessen nicht allein ein Resultat ihrer praktischen Erfahrungen, sie beruhten vielmehr sehr wesentlich auf theoretischen Arbeiten eines Kreises

[19] Eckert 1946a/b, bes. S. 531; ähnliche Überlegungen hatte Mauchly (1946, S. 25-39) in einem Vergleich von Analog- und Digitalmaschinen angestellt.

naturwissenschaftlich und philosophisch orientierter Mathematiker, die der eigentlichen Konstruktion ferner standen. Zu ihnen gehörten in erster Linie John von Neumann, Herman H. Goldstine und Arthur Burks, die seit 1944 an den Entwicklungsarbeiten der Moore School teilnahmen und die sich besonders intensiv an dem nachfolgenden EDVAC-Projekt engagierten. Erst ihnen gelang es, die Architektur-Erörterungen von der bisherigen Fixierung auf das Hardware-Engineering und der metaphorischen Bindung an frühere Einzweck- und Analogrechner endgültig zu lösen. Diese Expertengruppe, die an der Moore School bald als ″logicians″ den ″technologists″ gegenübergestellt wurde, entwickelte als Gegenkonzept zum amorphen ENIAC-Design und als systematisierende Verallgemeinerung des schon von Eckert und Mauchly anvisierten Stored-Program-Konzeptes für den ENIAC-Nachfolger EDVAC, ein ″complete logical framework for the machine″ (Goldstine 1972, S. 188, 192).

Vor allem John von Neumann stellte die ″design structure″ eines Einprozessor-Rechners als elementare Systemkonfiguration dar, als ein von Technologie und Hardware losgelöstes logisches ″system of organs″. Ihm ging es dabei nicht um konkrete Wirkstrukturen und Bauweisen von "units" und elektrischen Schaltkreisen, sondern um ein logisch funktionales Netzwerk von "Organen" und "Elementen", das sich als ein mathematisches Modell diskreter Gleichgewichte bzw. als Automatennetzwerk beschreiben ließ (v. Neumann 1945, S. 36). Während er für die Modellierung der Organstruktur eines Rechners und der Binnenstruktur der Organe noch die traditionellen Flußdiagramme verwendete, beruhte die Darstellung der dynamischen Rechenprozesse auf den Aktivitätsnetzen bzw. Ablaufdiagrammen des informationstechnischen Nervensystemmodells von Warren McCulloch und Walter Pitts. Deren Schaltwerkmodelle biologischer Neuronennetze wurden darüber hinaus das Vorbild für eine axiomatische Darstellung des Verhaltens der Grundeinheiten "künstlicher" und "natürlicher Automaten" und damit neben dem mathematischen Modell einer universellen infomationsverarbeitenden Maschine Alan Turings die zweite Säule der von v. Neumann ab 1948 in Angriff genommenen "General and Logical Theory of Automata" (v. Neumann 1951, S. S 104, vgl. Aspray 1990, S. 173 ff., 180 ff, 189 f.). Ähnlich wie bei Norbert Wiener die Regelungstheorie und bei Ludwig v. Bertalanffy die Systemtheorie, beruhte auch die erste wissenschaftliche Grundlegung von Rechnerarchitekturen auf einer wechselseitigen Analogiebildung zwischen informationstechnischen, biologischen und neurophysiologischen Strukturen und Prozessen. Durch anfangs bewußte, später auch unbewußte metaphorische Übertragungen kam es hier zu den auffällig biomorphen bzw. anthropomorphen Begriffen und Vorstellungen in dem

neuen Technikzweig, während auf der anderen Seite informationstechnische Begriffe und Funktionsmuster Eingang in die Biowissenschaften fanden (Schmidt-Brücken 1998).

Und doch ging es von Neumann bei der "neuron analogy" und Bezeichnungen wie "organ", oder "memory" nicht um die schlichte Behauptung tatsächlicher Übereinstimmungen "künstlicher" und "natürlicher Automaten", im Gegenteil, er betonte immer wieder die prinzipiellen Unterschiede in der Wirklich-keit. Der Modelltransfer und die wechselseitige Analogiebildung behielten vielmehr letztlich hypothetischen Charakter. Sie zielten auf eine systemische "logical and [...] organizational point of view" bzw. auf eine mathematische Beschreibung von logischen Systemstrukturen, die die Unmittelbarkeit der technischen Konstruktebene überwindet. Aber gerade mit dieser theoretischen Abstraktion stieß er auf Unverständnis bei den ENIAC-Erfindern: Eckert und Mauchly konnten dieser abgehobenen Betrachtungsweise der Organstruktur eines Computers und deren metaphorischer Verortung im Grenzbereich zwischen Informationstheorie, Physiologie und Thermodynamik nur wenig abgewinnen. Sie sahen darin mehr eine Übersetzung ihrer eigenen Erfindung in eine abstrakte, für die Konstruktion selber irrelevante Wissenschaftssprache. Von Neumann habe lediglich, so beklagte sich Mauchly später, die "modules" in "organs" umbenannt und aus den Elektronenröhren hypothetische Neuronen gemacht: "It was clear that Jonny was rephraising our logic, but it was still the *same* logic. Also, he was introducing different but equivalent symbols, nevertheless, the devices still did the same things."[20] Sie verkannten damit die eminent praktische Relevanz einer Theoretisierung der Strukturen des Rechners und des Rechnens.

Der Rückgriff auf die technologieunabhängige Notation der Neuronennetzwerke ermöglichte es v. Neumann, den Computer durchgängig als ein System aus Black-box-Beziehungen zu formalisieren. Die automatentheoretische Sichtweise lenkte die Aufmerksamkeit zudem stärker auf das dynamische Prozeßverhalten einschließlich der damit verbundenen zeitökonomischen Aspekte als es die zeitlose Betrachtungsweise des Logikkalküls vermocht hätte.[21] Von

---

[20] Mauchly 1979, S. 217; vgl. auch Eckert, J. P.; Mauchly, J. W., Automatic High-Speed Computing: A Progress Report on the EDVAC, University of Pennsylvania, Philadelphia, 30. September 1945, zit. nach: Aspray 1990 , S. 42, ("In his report, the physical structures and devices proposed by Eckert and Mauchly are replaced by idealized elements to avoid raising engineering problems which might distract attention from the logical considerations under discussion."); vgl. auch das Goldstine-Briefzitat bei Aspray 1990. S. 41.

[21] Vielleicht erklärt sich aus der Differenz von logischer und automatentheoretischer Betrachtungsweise auch die stark abweichende Einschätzung der Zeitökonomie bei Zuse und von Neumann, siehe dazu oben den Beitrag von H. Petzold.

einer Weiterentwicklung der noch im Stadium eines kaum systematisierten 'Erfahrungsschatzes' befindlichen Automatentheorie zu einer "detailed, highly mathematical, and more specifically analytical theory of automata and of information" erhoffte er sich darüber hinaus wesentliche Verbesserungen der Zuverlässigkeit von Rechnern, insbesondere bei komplexeren Strukturen (v. Neumann, 1951, S. 101; 1960, S.14). Ob er damit auch das Fernziel verfolgte, das Rechnerdesign zu einer streng wissenschaftlichen Disziplin zu machen, erscheint jedoch fraglich. Denn seine Traktate über den Rechner- und Programmentwurf aus den Jahren 1945-54 legen eher den Schluß nahe, daß er von einem Zusammenspiel von theoretischer Reflexion und erfahrungsbasierten Organisationsleistungen im konstruktiven Prozeß ausging. Da die axiomatische Modellbildung der "Automatentheorie" offenbar nicht alle Vorgänge im Rechner und alle Entscheidungen beim Design abdeckte, insbesondere nicht die Nutzungs- und Ökonomie-bezogenen, rekurrierte er in den Memoranden immer wieder auch auf konstruktives Erfahrungswissen. Durch die Kombination beider Vorgehensweisen gelangten er und die anderen "logicians" in den Architekturtraktaten einerseits zu theoretischen Verallgemeinerungen, die den reinen Empirikern nicht möglich waren, die sich aber für die funktionale und konstruktive Strukturierung von Computern und selbst für Dimensionierungs-Entscheidungen als äußerst folgenreich erweisen sollten. Andrerseits erzielten sie Einblicke in konkrete Strukturentscheidungen und die Organisation von Designprozessen, die bei den späteren rein szientisti-schen Architekturkonzepten meist ausgeblendet blieben.

Schon im "First Draft" und der "Preliminary Discussion of Logical Design" ist das produktive Wechselspiel von mathematisch-logischer Formalisierung und systematisierter Empirie deutlich sichtbar. So wurde dort die elementare Organstruktur mit ihrer minimalen Konfiguration und äußersten apparativen Vereinfachung durch eine strikt sequentielle Organisation auf *logischer* Ebene zwar gleichsam als ”One-best-way-Lösung” gedacht, doch sahen die Autoren zugleich ganz klar, daß auf der Basis der Kernorgane sehr verschiedene Bauweisen und Funktionsabläufe möglich sind, oder wie Goldstine es später formulierte: ”Many different machine organizations and architectures are possible with five organs of von Neumann’s account” (Goldstine, 1972, S. 204). Douglas R. Hartree (1947, S. 56) brachte hier die Unterscheidung einer anatomischen Sicht des Rechners als *Struktur* von Organen und der physiologischen Betrachtung der *Funktions*beziehungen zwischen ihnen ins Spiel. Die Vielfalt der möglichen Lösungsvarianten und vor allem die starke Interdependenz der Systemkomponenten machten die Rechnerkonfiguration auf der Designebene

damit für von Neumann bereits zu einer komplexen Problemstruktur, deren Zielkonfliktcharakter kein sequentielles Vorgehen und Abhandeln erlaubte.[22]

Es war schon deshalb keine saubere Separierung von Aufgabenpaketen möglich, weil ″discussions of specific parts will be mixed with discussions of the general principles″. Die argumentative wie designmäßige ‘Architektur’-Entwicklung ist deshalb nicht von einem linearen Top-down-Prozeß, sondern von mehrfachen Iterationen und stufenweiser Annäherung der ″arrangements″ an die ″desired features″ geprägt, ein Zickzack-Verfahren, bei dem sich die Aufgaben- und Einsatz-spezifische Lösung herauskristallisiert, die auch ökonomischen Optimalitätskriterien entspricht (ebda.). Zielkonflikte sehen die Traktate der von Neumann-Gruppe auch bei der Wahl und Ausgestaltung der "features" eines Rechners: es müsse ständig zwischen einem zu viel und einem zu wenig, zwischen "radicalism and conservatism" abgewogen werden: "We wish to incorporate into the machine - in the form of circuits - only such logical concepts as are either necessary to have a complete system or highly convenient because of the frequency with which they occur and the influence they exert in the relevant mathematical situations." (Burks, Goldstine, v. Neumann 1946, S. 97) Das entscheidende Designkriterium für die Funktionalität war für sie letztlich ein "consistent and sound principle". Damit wurde hier neben dem Zielkonflikt-Modell auch bereits das Konsistenzprinzip in die Architekturdebatte eingeführt, das Frederick Brooks später zum Kern seines Architektur-Konzeptes erhob.

Auch alle Dimensionierungs-Entscheidungen im Rechnerdesign wurden einer quasi thermodynamischen Betrachtung unterworfen. Die Dimensionen der ″specific parts″ müßten die "limiting factors in any digital machine" berücksichtigen und den Systemcharakter der Organstruktur beachten, denn sonst bestehe die Gefahr einer Störung der ″balance of the machine″ (v. Neumann 1946, S. 247 f.). Aus der Perspektive eines systemischen Gleichgewichtsmodells betonte von Neumann die Unsinnigkeit einer einseitigen Beschleunigung von Teilprozessen, da dies nur zu Engpässen bei langsameren Prozessen und damit zu Ungleichgewichten im Gesamtsystem führen müsse, eine Einsicht, die, obwohl von Edmund C. Berkeley (1956, S. 141f.) zum Prinzip des "balanced design" zusammengefaßt und im Amdahlschen Gesetz noch einmal bekräftigt, von

[22] "The ideal procedure would be, to take up the five specific parts in some definite order, to treat each one of them exhaustively, and go on to the next one only after the predecessor is completely disposed of. However, this seems hardly feasible. The desirable features of the various parts, and the decisions based on them, emerge only after a somewhat ziggzagging discussion" (v. Neumann 1945, S. 36).

vielen Computerarchitekten immer wieder mißachtet wurde (v. Neumann 1946, S. 247 f.). Wohl im Hinblick auf das komplexe Zusammenspiel von Struktur- und Dimensionierungs-Entscheidungen und der ständigen Abwägung von generellen Designprinzipien wie z.B. "desire for simplicity, or cheapness of the machine" und organspezifischen "principles" sah von Neumann den Entwurf des Ganzen und der Einzelorgane ausdrücklich als einen Vorgang der "organization"[23]

Der Organisationsbegriff spielt überhaupt in den Traktaten von 1945/46 eine wichtige Rolle, ein Sachverhalt, der wegen der Fixierung auf die Neuronen-Metaphern bisher wenig beachtet wurde. In einer späteren Zusammenfassung seines Architekturkonzeptes vor deutschen Computerpionieren im Jahre 1955 rückte von Neumann (1955, S. 7 ff. 12 ff., 18, 26) den Organisationsbegriff sogar ganz in das Zentrum: Das Ziel des Entwurfs ist eine "gut organisierte Maschine"und die ist das Resultat einer Abstimmung der "wesentlichen Organisation einer Ziffernmaschine" (d. h. eines Digitalrechners) mit der Organisation der einzelnen "logischen Organe" und Ablaufprozesse, wobei neben "Operationsgeschwindigkeiten" und Kostenfaktoren die "Bedienungsorganisation" besonderes Gewicht hat. Der Organisationsbegriff wurde auch von Aiken, Stibitz, Hartree, Booth und anderen frühen Computerentwicklern für das Gesamtdesign verwendet, er wurde dann in den 50er Jahren so selbstverständlich, daß er später als "computer organization" neben die "computer architecture" zum zweiten zentralen Begriff dieser Technikdisziplin aufstieg, wobei die - keinesfalls durchgängige – Eingrenzung auf die innere Hardwarestruktur eines Rechners erst im Laufe der 70er Jahre erfolgte.

Mit der Darlegung des grundsätzlichen Zielkonfliktcharakters des Rechnerdesigns und des Kompromißcharakters der Designziele Geschwindigkeit, Verfügbarkeit, technischer Aufwand und Kosten sowie der wechselseitigen Näherungsvorgänge bei den Designkriterien entwickelte von Neumann zusammen mit den anderen ”logicians” bereits wesentliche Momente des Computerarchitektur-Konzeptes: eine auf Einfachheit und ”logical design” orientierte elementare Organstrukturlehre und eine bereits in Grundzügen erkennbare Designmethodik, die ansatzweise auf einer interdependenten Problem- und Trade-off-Struktur basierte. Sie suchten auch schon nach Bezeichnungen für diese übergeordneten Strukturmerkmale und Designaspekte und verwendeten

[23] V. Neumann 1945, S. 40 ("the device should be as simple as possible, that is, contain as few elements as possible"), 54, 57, 66; Burks, Goldstine, v. Neumann 1946, S. 92

hierfür vor allem Begriffe wie "wesentliche Organisation", "logical structure", "logical design" und "design structure" bzw. "structure of a computing system". Mit ihren Architekturtraktaten von 1945 – 1948 definierte diese Gruppe nicht nur die eigentliche Kernstruktur der bis heute dominierenden Computerarchitektur, sie legte auch die Basis für das *theoretische Konzept* der Architektur, für die Struktur- und Bauformenlehre und damit den Ausgangspunkt für die Herausbildung einer Technikwissenschaft und Designlehre im Computer-bereich. Die Traktate bestimmten daher auch bald die Architekturdebatte in der Computer Community. Selbst Pioniere der ersten Stunde, wie Alan Turing und George Stibitz und, wie bereits erwähnt, John Presper Eckert, bekamen durch sie wesentliche Anstöße zu einer theoreti-schen Begründung oder Reformulierung der eigenen Design-anschauungen.

Alan Turing, der seinerseits mit seinen frühen theoretischen Arbeiten über universelle Automaten John von Neumann stark beeinflußt hatte, erhielt durch dessen Traktate entscheidende Anregungen für seinen eigenen Entwurf eines Stored-Program-Computers am National Physical Laboratory an der Wende 1945/46. Turings "Proposal for Development in the Mathematics Division of an Automatic Computing Engine (ACE)" übernahm nicht nur den Organbegriff von Neumanns, die Struktur der "main compo nents" und die auf McCulloch und Pitts zurückgehende Notation für die Elemente-Beziehungen und Rechenprozesse, sondern enthielt auch direkt die Empfehlung, den Text in Verbindung mit von Neumanns "Report on the EDVAC" zu lesen (Turing 1946, S. 21). In seinem Traktat setzte Turing jedoch durchaus eigene Akzente, er rückte die Speicherkapazität, wie bereits in seinem Denkmodell für Universalmaschinen von 1936, in das Zentrum seiner "main strategic ideas behind digital computing machinery": "I believe that the provision of proper storage is the key to the problem of the digital computer [...]. In my opinion this problem of making a large memory available at reasonably short notice is much more important than that doing operations such as multiplication at high speed. (Turing 1947, S. 112 f.). Für Turing war die "computing engine" nämlich nicht wie für von Neumann vorrangig ein mathematische Aufgaben lösender Automat mit maximalem Durchsatz, sondern bereits eine Informationen beliebiger Art verarbeitende "universal machine", die er als ein möglichst großes logistisches Speichersystem und nicht als eine thermodynamische Strömungsmaschine betrachtete. Turing betonte auch noch mehr als von Neumann die Abhängigkeit der Dimensionierung der Hauptkomponenten des Rechners vom jeweiligen Verwendungszweck: Die Zielkonflikte zwischen der Schnelligkeit der

Recheneinheiten, dem Speicherumfang und der Zugriffsdauer variieren je nach kommerzieller oder wissenschaftlicher Nutzung.

Schließlich wurde in Turings "basic design"' die Darstellung der elementaren Kernstruktur stärker in eine Folge methodischer "design steps" integriert, d. h. in die Kette von der "composition of the calculator", der Strukturierung und des Entwurfes der "basic components", der Festlegung der Kontrollstruktur, der Gestaltung der externen Organe (Input/Output), der Einengung der "scope of the machine", der Wahl der Programmierungstechniken, der Testphase und schließlich der Gestaltung der Betriebsbedingungen (ebda., S. 21 ff.). Das Designphasenmodell war damit bereits rechnerspezifisch strukturiert und nicht mehr, wie noch bei Aiken, vom Vorbild der Elektrokonstruktion bestimmt. Turing perfektionierte durch mehrfache Wiederholung des Designs in einer Reihe von Papierversionen seine Designmethodik, sodaß er innerhalb kürzester Zeit nach Abschluß einer Version eine neue schaffen konnte (Williams, 1985, S. 337). Obwohl er mit seinem systematischen Vorgehen ohne physikalische Umsetzung die anderen Designer frustrierte und einen erheblichen Zeitverzug und schwere Projektmanage-mentprobleme heraufbeschwor, legte Turing mit seinem Vorgehen den Grundstein für ein Rechnerdesign, das schneller war als die vergleichbaren Systeme in Manchester und Cambridge und trotzdem mit weniger Hardwareressourcen auskam. Doch seine Strategie, auf der Hardwareseite mit einer "simple machine" zu arbeiten und dafür das "coding" zu perfektionieren, wurde auf Dauer kein Erfolg, da sich hierdurch die Systemkomplexität deutlich erhöhte (Huskey, 1948/49; Goldstine, 1972, S. 218).

George R. Stibitz gelangte offensichtlich durch die Bekanntschaft mit von Neumann seit April 1944 und durch die Traktate der Von-Neumann-Gruppe zu einer systematischeren Sicht des Rechnerdesigns (Aspray 1990, S. 32 f.). Er hatte bei den Bell-Laboratorien bereits in den 30er Jahren Spezialrechner auf Relaisbasis für die Berechnung komplexer Zahlen und für ballistische Berechnungen entworfen, auf deren Grundlage dann 1947 ein "all-purpose computing system" entstand. Noch in seinem ersten Memorandum "Computer"' von 1940 war er kaum über das engere Logikdesign hinausgegangen, d. h. über die Umsetzung von Logik- und Rechenprozessen in den Schaltungsentwurf. Die ersten öffentlichen Beschreibungen der Bell-Rechner von 1946 bis 1948 standen bereits unter dem Einfluß der Von-Neumann-Traktate. Zwei Aufsätze von Stibitz aus den Jahren 1947/48 nahmen dann das von Neumannsche Architekturkonzept zum Ausgangspunkt für eigene grundsätzliche Erwägungen über die "Organization of Large-Scale Calculation Machinery" sowie über

Planungs- und Designprinzipien von "automatic computers". Stibitz folgte dem Organstrukturmodell und den Balancing- und Dimensionierungs-Überlegungen von Neumanns, ebenso den biomorphen Analogien mit dem Nervensystem und dem Gehirn. Doch gelangte er durch eine stärkere Einbeziehung der eigentlichen Nutzer von Rechenanlagen zu anderen Folgerungen für die Größendimensionierung von Rechen-anlagen, zu einer erweiterten Sicht der Zielkonfliktstruktur des Designs und am Ende sogar zu einem komplexeren Architekturverständnis.

Während von Neumann (1946, 1949) mit Blick auf die serielle Arbeitsweise des Ein-Prozessorcomputers auf möglichst hohe Arbeitsgeschwindigkeiten setzte und dabei das "economical optimum" in "large computing machines" sah, strich Stibitz stärker die Nachteile von Großanlagen für die Nutzer von Rechenleistungen heraus: Der "traffic factor" bei vielen Nutzern führe zu mehr "routing problems", zu mehr Blockaden und Verfügbarkeitsproblemen, wobei beim Ausfall einer zentralen Anlage immer gleich alle Nutzer betroffen seien. Am nachteiligsten erschien ihm aber, daß die Berechnungen nicht mehr, wie bei einer kleinen Anlage, von den Programmierern selber oder "in close cooperation between the operator of a computer and the person or group with which the problem arise" bearbeitet würden, sondern daß alles nur noch über einen "staff of operators" laufe. Stibitz sah damit bereits 1947, d. h. noch vor der Etablierung des Rechenzentrums- und Batchbetriebes, deren gravierende Schwächen voraus. Im Gegensatz zur Größerdimensionierungs-Strategie, die von Neumann der IBM als Leitlinie empfahl und die auch bald die Entwicklungsrichtung des Mainframe-Computing bestimmte, forderte Stibitz als einer der ersten möglichst kleine, Aufgaben-angemessene Rechner in der Verfügbarkeit der Nutzer: " [...] the originator of a problem should have a machine available for his own use [...] the writer's opinon is that automatic computers should be designed in the smallest units consistent with the problems to be handled." (Stibitz 1947, S. 142)

Die Einbeziehung der Nutzer und der Nutzungseigenschaften veränderte damit aber auch die in den Von-Neumann-Traktaten entwickelte Zielkonfliktstruktur beim Design, denn zur Geschwindigkeit, zum Hardwareaufwand, zu den Erstellungskosten, den Personalressourcen der Bedienorganisation kamen nun der Gebrauchswert und die Wartungs- und Betriebskosten hinzu. Damit aber traten neben die quantifizierbaren Fließ- und Gleichgewichtsgrößen des informationstheoretisch-thermodynamischen Bilanzierungsmodells von Neumanns auch qualitative Bewertungsaspekte, die die Designkomplexität deutlich

erhöhten. Implizit erkannte Stibitz hier bereits, daß die Ausdehnung der Designerperspektive auf einen größeren Ausschnitt des Produktlebenszyklus die Zielkonflikte und die Kompromißbildung verschiebt und erschwert. Er trug dieser gestiegenen Designkomplexität dann auch dadurch Rechnung, daß er das Gesamtdesign in zwei große Aufgabenpakete aufteilte, die ‚external organization', die das Design mit den Nutzern klären und abstimmen muß, und die ‚internal organization', die die Bau- und Kontrollstruktur der Maschine festlegt. Während letztere hauptsächlich in der Verantwortung des Designers liegt, hat er über die externe Organisation keine Kontrolle, hier bestimmen die "agencies" und das "environment" der Nutzung die Designvorgaben (Stibitz 1948, S. 91 ff.).

Stibitz löste sich damit von der vorherrschenden Sicht eines vorwiegend durch die mathematische Logik und Technik determinierten "logical design" und wies dem Entwickler lange vor Frederick Brooks' Architekturkonzept eine Mittlerrolle zwischen den Benutzeranforderungen und den technischen Designentscheidungen zu. Die Strukturbildung wurde so über die technische Problemlösung hinaus zu einer vorrangig organisatorischen Aufgabe, zur wirklichen "organization of the computer". Die Multiperspektiven-Sicht ließ Stibitz aber auch zu einem neuen Verständnis der internen Organisation des Rechners gelangen. Er unterschied zwei grundlegende "general plans of the internal organization" (ebda., S. 96): einen automatenartigen Typus mit festem Zyklusablauf und einfachem Kontrollmechanismus und einen netzartigen Typus mit wahlfreien Kommunikationsflüssen. Letzteren sah Stibitz wie von Neumann in Analogie zum Nervensystem, doch ging es ihm dabei nicht nur um eine logisch-abstrakte Darstellung von Rechnerprozessen in Gestalt von Ablaufdiagrammen, sondern darüber hinaus um eine Interpretation der Vorgänge in einem Rechner als Kooperation von Instanzen mit unterschiedlicher "intelligence".

Stibitz gliederte die Gesamtaufgabe des Rechners in eine Kette kommunizierender Prozesse, eine "series of levels of intelligence" (ebda., S. 96-98). Der oberste "intelligence level" bildet die "over-all control" des vom Operator vorbereitetes Programmablaufs, die zweite Ebene stellt die für die Ausführung notwendigen Instruktionen und Subroutinen bereit, die dritte die Elementaroperationen und die vierte schließlich die Relais- und Röhren-Schaltungen, die die Operationen physikalisch umsetzen. Der Computer wurde damit nicht mehr als elementare Organstruktur, als thermodynamisches Verhaltensmodell oder elektrischer Schaltplan dargestellt, sondern als eine Hierarchie kooperierender Schichten. Stibitz war sich der Außergewöhnlichkeit seines Ebenenmodells

bewußt, er begründete es auch ausdrücklich damit, daß man mit ihm der gestiegenen Designkomplexität adäquat begegnen könne: ”It is clear that the division of the computer into levels of intelligence is not a necessary condition for a computer. It is merely *a way of thinking about the rather complex problems* presented by a difficult situation. It has, however, been found to be of considerable help in this capacity. It leads to economy of equipment, since, wherever possible, repeated operations are carried out by the same equipment without calling for elaborate instructions from the operator. And, as an important contribution, it makes the *plan of the machine* relatively easy for maintenance people to understand. It simplifies testing of the components since each level may be isolated and tested separately” (ebda., S. 97, meine Hervorhebungen). Damit wurde schon im Jahre 1947 zum ersten Mal ein Schichtenmodell für Rechnerstrukturen eingeführt und, wie fast zwei Jahrzehnte später im Softwarebereich bei Dijkstra und Parnas, wurde die saubere Funktionstrennung der einzelnen ”level” mit der Erleichterung arbeitsteiliger Entwicklungs-, Test- und Wartungsprozesse begründet. Der soziale Prozeß der Rechnerkonstruktion fand damit einen Niederschlag in der soziomorphen Modellierung der Strukturen und Prozesse eines Computers. Stibitz griff damit so weit voraus, daß außer Maurice Wilkes ihm kaum ein Computerarchitekt der Zeit folgen mochte. Erst nachdem das hierarchische Ebenenmodell und ”Hiding”-Prinzip in den 60er Jahren in der Softwarekonstruktion neu entwickelt worden waren, fanden sie von dorther eine breiteren Eingang in die Rechnerarchitektur.

Stibitz war nicht der einzige Computerpionier, der den Rechner als ein quasi sozial strukturiertes Gebilde ansah. Lange vor ihm hatte schon Babbage seine Difference und Analytical Engine als eine industrielle Fortführung der manufakturellen Tabellenberechnung Gaspard de Pronys entwickelt und die Teile der Maschine entsprechend als mechanisiertes System der Arbeitsteilung der Fabrik interpretiert, als "mill", "store" und "directive part"(Babbage 1835, S. 191-202). Einige Jahre nach dem hierarchischen Kooperationsmodell einer Rechnerarchitektur bei Stibitz findet sich bei dem deutschen Pionier elektromechanischer Rechenautomaten Alwin Walther ein dezidiertes Herrschaftsmodell eines Computers. Walther, der 1942-44 mit Hilfe von steckbaren "Kopplungstafeln" übliche elektromechanische Rechenwerke, Tabulatoren bzw. Lochkartengeräte zu programmgesteuerten Rechenautomaten verband, interpretierte den "Aufbau von Rechenautomaten" in Vorträgen von 1946, 1952 und 1955 als Muster und Kernstück der vollautomatischen Fabrik. Die Kopplungstafel fungiert danach als "Befehls-Steuerwerk", das die "Einzelwerke" steuert, bzw. als "Gehirn", das "Arbeitsbefehle" an die Glied-maßen und Muskeln" gibt, d.h. die Addier- und

Multiplizierwerke (Walther 1956, S. 17). Letztere verglich er auch mit dem Fließband, während das "Kommandowerk" das "Analogon zur Betriebsleitung" darstelle (ebda., S. 38). Im Unterschied zu der bei von Neumann und Stibitz noch weitgehend bewußten und hypothetischen Analogiebildung wird der soziomorphen Metaphorik bei Walther Wesenscharakter zugewiesen: Die gleichzeitige Konstruktion des Rechneraufbaus als Abbild der sozialen Herrschaftsorganisation und die Deklarierung des Rechners als logisches Strukturmuster für die Fabrikorganisation lassen Ursprungs- und Zielbereich der Metaphernbildung verschwimmen. Die soziomorphe Modellierung von Technostrukturen und die technomorphe Modellierung von Sozialstrukturen verfestigen sich wechselseitig und erhalten dadurch Ideologiecharakter.

Im Laufe der Pionierphase der Computertechnik zwischen der Mitte der 30er und dem Ende der 40er Jahre waren so eine Reihe zum Teil recht verschiedener und heterogener Architekturkonzepte entstanden. Generell waren dabei Struktur- und Bauformen-Erörterungen noch nicht von Konstruktionsempfehlungen und Designlehren bzw. Designphasenmodellen geschieden. Für die Modellbildungen wurden, wie für neue Technikzweige üblich, Anleihen bei etablierten Technikwissenschaften gemacht, wie es vor allem in den Flußdiagrammen bzw. Schaltplan-Darstellungen und Designphasenmodellen zum Ausdruck kommt. Daneben wurden Analogien aus der Mathematik und den Naturwissenschaften herangezogen, um die neuartigen infor-mationstechnischen Gebilde zu modellieren, so besonders aus der Thermodynamik und Kybernetik, Anatomie, Physiologie und die Neuronennetzwerke der Neurophysiologie. Schließlich finden sich neben den Fließ- und Schaltmodellen auch soziale Organisationsmodelle als Strukturanalogien für die ‘Arbeitsteilung’ technischer Komponenten im Aggregat und als Hinweis auf die akteursbezogene und konfliktbehaftete Allokationsproblematik bei Hardware- und Programmier-Ressourcen im Gesamtprozeß des Computing. Dieser ausgeprägte Pluralismus wie der stark metaphorische Charakter der Architekturmodelle war typisch für eine Technikwissenschaft in einem sehr frühen Stadium, und doch hatte die Erörterung der Strukturformen und Bauweisen hier noch vor der eigentlichen Innovationsphase bereits ein Reflexionsniveau und einen Systematisierungsgrad erreicht, wie ihn kaum eine andere Technikdisziplin aufzuweisen hatte. Mit dieser fortgeschrittenen Theoretisierung vor und neben der Technikentwicklung steht der Computerbereich aber in gewissem Widerspruch zu den wissenschaftssoziologischen Entwicklungsmodellen.

Empiriebasierte Architekturkonzepte der Innovationsphase

Vergleichende Auswertung realisierter Lösungsmuster: Forrester 1951-55

- Systematische Vergleichsbetrachtungen zu den "physical techniques", "design procedures" und "design stages"
- Problem der Vergleichsmaßstäbe für die Struktur- und Designbewertung: Forderung nach "more sophisticated criteria of comparison"

Stilvarianten- und Komplexität-reduzierendes One-best-way-Designkonzept: Wilkes, 1951

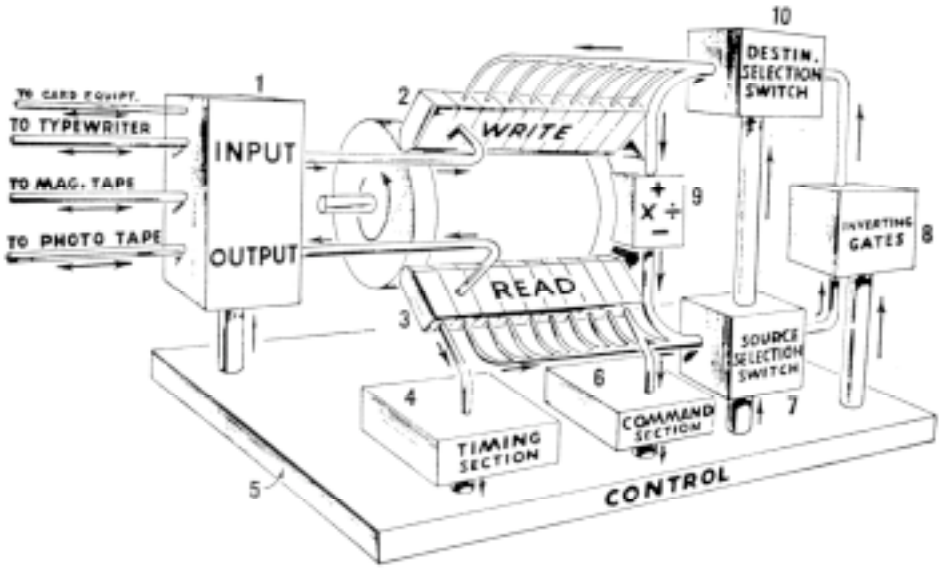
- Erklärung von Stilunterschieden im Computerdesign aus unterschiedlichen "constraints" der Designteams (Zeitpunkt und Größenordnung des Projekts, Technologie-verfügbarkeit, Vorerfahrungen, disziplinäre Zusammensetzung)
- Suche nach einem "best way to design an automatic calculating machine": Vermeiden von "ad hoc manners" und heterogenen "cross connections" im Design
- Reduktion der Maschinenkomplexität durch viele Wiederholteile und eine durch Blockstrukturen vereinfachte Verknüpfung der Rechnerkomponenten und Mikroprogrammierung

Bewußte empirische Designmethodik auf komparatistischer Basis: Rutishauser/Speiser 1950, Speiser 1957

- Systematische Vergleiche unterschiedlicher Computerbauweisen
- Explikation der Designphilosophie als bewußte Strukturentscheidung und Lösungskombination auf breiter Vergleichsbasis
- Erfahrungsbasierte Fortführung der systemischen Sicht des Rechners bei v. Neumann: das Gleichgewichtsmodell und die Zielkonflikt-Analyse als Angelpunkt ausgewogener Dimensionierung

Erste Systematisierungen der Designerfahrungen und Strukturvergleiche in Lehrbuchform: Wilkes, 1956, Phister, 1958, Flores, 1960

- Systematische Zusammenstellungen des "fund of 'unwritten art' in the computer field": Systematik der Funktionen und Bauteile, Zusammenstellungen von Lösungsvarianten, kritischen Engpässen und Irrwegen, Vorteil-/Nachteil-Bilanzen einzelner Bauformen und Lösungsmuster, Beschreibungen von Designkonflikten, Formulierung übergeordneter Designprinzipien
- Systemtechnische Strukturierung der Anlagenkomplexität ("modular construction") und der Designphasen nach Vorbild des "Systems Engineering" (Phister, Flores)
- Hierarchische Strukturierung des gesamten Hardware-/Software-Systems: "hierarchy of programming" und Einteilung der "structure of a computer" in hierarchische "levels of organization" (Flores)



Maschinenmodell der Rechnerarchitektur (Bendix-G5, Huskey, Korn 1969, S. 20-16)

3. Empiriebasierte Architekturkonzepte der Innovationsphase: Vom Anlagenvergleich zu Anläufen für ein Computer Engineering

Auf die Periode vorempirischer Design- und Architekturreflexionen folgte seit Ende der vierziger Jahre eine Learning-by-doing-Phase, in der grundsätzliche Erörterungen etwas in den Hintergrund traten. Bei den Einzelentwicklern und den Hochschulteams stand nun die Realisierung von funktionstüchtigen und zuverlässigen Anlagen im Mittelpunkt. Man griff dabei oft Designideen und Problemlösungen vorhandener Maschinen bzw. paralleler Entwicklungen auf, gelangte aber zunächst nur selten zu systematischen Vergleichen der Bauweisen. Erste Gelegenheiten für komparatistische Betrachtungen boten die Ende der vierziger Jahre aufkommenden Computerkonferenzen in den USA und England, etwas später auch in Deutschland und Frankreich. Bei diesen Treffen der Communities prallten durch die wechselseitige Präsentation der Projekte und Anlagen die unterschiedlichen Bauformen und Designphilosophien aufeinander. Die abweichenden Terminologien und Notationen machten dabei die Verständigung zwischen den verschiedenen Forschergruppen oft sehr schwierig. Die Begriffssprache war noch stark von dem jeweiligen lokalen „laboratory slang“ bestimmt und entsprechend uneinheitlich. Die meisten verwendeten zwar die Organbezeichnungen von Neumanns, doch verstand man z. B. unter ”control” oft sehr Verschiedenes, ebenso unter ”programme” bzw. ”program”,

unter ”coding” oder "parallelism”. Die gleiche Sprachverwirrung herrschte bei den Blockdiagrammen, da jede Forschergruppe trotz großer Familienähnlichkeit der Notationen eigene Varianten entwickelte (Wilkes 1956, S. IX). Besonders die nationalen und später internationalen Konferenzen wurden deshalb Orte bzw. Ausgangspunkte für die Normung der Fachsprache des entstehenden Technikzweiges (Hartree 1954, S. 217 f.).

Obwohl die Erfinder und Entwicklerteams ihre Anlagen oder Projekte meist nur aus der eigenen Perspektive darstellten und bewerteten, führte allein schon die bloße Gegenüberstellung der verschiedenen Ansätze zu Erkenntnissen über die Architektur- und Designdifferenzen. Daraus ergaben sich erste Vorteil-Nachteil-Bilanzen unterschiedlicher Lösungen, ebenso Hinweise auf Irrwege und kritische Engpaßfaktoren im Design. Grundsätzlichere Unterschiede in Bauweisen wurden oft in Form von z. T. heftigen Systemstreiten[24] ausgetragen, so besonders bei den folgenden Systementscheidungen:

- „general purpose“ oder „special purpose“- Systeme,
- digitale oder analoge Betriebsweise,
- integrierter oder getrennter Befehls- und Zahlenspeicher
- parallele oder serielle Wortverarbeitung,
- sequentielle oder parallele Befehlsausführung,
- binäre oder dekadische Zahlendarstellung,
- Fließkomma- oder Festkomma-Arithmetik
- Ein- oder Mehradressen-Computer,
- zentrale oder verteilte Register-Organisation, [25]
- asynchrone oder synchrone Systeme,
- „single data bus“ oder „multiple bus organization“ sowie
- feste oder variable Wortlänge

Während sich einige Systemstreite relativ schnell erledigten – so bei den externen Speichern und bei der Zurückdrängung der Spezialrechner und Analogsysteme zu Nischentechniken – zogen sich andere noch weit in die 50er und z.T. sogar in die frühen 60er Jahre hin. So hielt selbst von Neumann (1955, S. 10 f.) den Systemstreit zwischen Binär- und Zehnersystem noch 1955 für nicht endgültig entschieden und die IBM baute noch zwischen 1957 und 1960

---

[24] Vgl. hierzu bes. Brainerd, Sharpless 1948, S. 1032 f.; Sobel 1970, S. 197-202 ; zum Konzept des Systemstreites siehe Hellige 1993, S. 205-210

[25] Andrews, Arsenault, Huskey 1962, S.20-5-7 sprechen hier von der Kontroverse "Grand Central Station vs. Source-Destination".

transistorisierte Versionen ihrer Dezimalmaschinen 620, 650, 702 und 705 (Bashe u. a., 1986, S. 439 f.). Nicht nur in erster Linie kommerzielle Rechner, sondern auch die 1954 bzw. 1960 fertiggestellten Hochleistungsmaschinen NORC von IBM und LARC von Sperry Rand arbeiteten noch im Dezimalprinzip und auch die Designer des IBM-Stretch-Computers begründeten noch 1959 bzw. 1962 den Bau eines Hybridsystems mit den großen Vorteilen des Dezimalsystems für den Nutzer (Buchholz, 1962, S. 43 f.; bes. die Tabelle auf S. 274). Auch das Floating-point-Prinzip setzte sich erst sehr langsam durch. Vor allem bei vielen kleineren und mittleren Rechnern, so belegen es vor allem die Computerkonferenzen, hielten sich von dem ‚klassischen' Von-Neumann-Typ abweichende Architekturformen bzw. Merkmalskombinationen noch recht lange.[26] Die vom Kuhnschen Modell her abgeleitete Annahme der Durchsetzung eines gültigen Paradigmas zu einem festen Zeitpunkt stimmt mit der tatsächlichen Anlagenvielfalt nicht überein.

Neben prinzipielleren Differenzen wurden bei den Vergleichen aber auch variierende Schwerpunktsetzungen bei den Designmerkmalen wie Geschwindigkeit, Bedienungsfreundlichkeit (serviceability) Verfügbarkeit (reliability), Zuverlässigkeit (consistency or uniformity in action) und Wartbarkeit sichtbar. Während für viele Entwickler wie für von Neumann die Geschwindigkeit das herausragende Designkriterium bildete, setzten andere mehr auf "circuit simplicity and reliability" oder Bedienungsfreundlichkeit als "design philosophy" (Wilkes 1954, S. 229; Cooke-Yarborough S. 465). Bei dem Für und Wider für einzelne Gestaltungsmerkmale wurde einerseits deutlich, daß sich bei unterschiedlichen Nutzerpopulationen und Anwendungsschwerpunkten auch abweichende Merkmalsprioritäten ergaben, andrerseits wurde sehr oft auch der durchgängige Zielkonfliktcharakter der Designkriterien angesprochen. So resümierte Werner Buchholz (1953, S. 1263), der „Senior Systems Planner" der IBM, über die Zielkonflikte bei der Wahl der "features" und Eigenschaften des Systemdesigns: "Many of these choices were interrelated, so that designing the system took on some of the aspects of solving a giant jigsaw puzzle". Der "complex nature of the puzzle" könne man nur mit einem möglichst einfachen „logical design" begegnen. Der DEUCE-Designer A. C. D. Haley von English Electric kam ebenfalls zu dem Resultat, daß das Design eines elektronischen Computers einen Kompromiß zwischen Verarbeitungsgeschwindigkeit, Spei-

[26] Die Annahme von Furger, Heinz 1998, S. 240 , um 1950 habe sich die von-Neumann-Architektur mit der später gültigen Kombination von Designmerkmalen bereits voll durchgesetzt, trifft jedenfalls nicht zu, die ERMETH war zu ihrer Zeit keineswegs eine Ausnahme.

cherkapazität und Equipment-Ökonomie darstelle: „All these aspects are mutually conflicting, and all have some influence on the important question of reliability" (Haley 1956, S. 165).

Grace Hopper und John Mauchly (1953, S. 1250) betonten besonders die Zielkonflikte zwischen Computerdesign und Programmtechniken. Setze man im Interesse niedriger Kosten vor allem auf "simplicity of equipment", so erhöhe sich dadurch meist der Programmieraufwand. Bewirke ein zusätzlicher Hardware-Einsatz aber höhere Verarbeitungsgeschwindigkeit, so könne dies den Betrieb insgesamt verbilligen: "The design of the general-purpose computer must always be a compromise involving many different requirements which are to some extend conflicting". Langfristig hofften sie aber auf eine gewisse Konvergenz zwischen "computer design" und der "art of programming". Angesichts dieser unterschiedlichen Designkriterien und permanenten Designkonflikte kam der Chefdesigner des Whirlwind I, Jay Forrester, 1951 (S. 109 f.) auf der nationalen Computerkonferenz zu dem Resultat, daß man sich in einem Bewertungsdilemma befände. Auf der einen Seite reichten Geschwindigkeit und Speicherraum als Vergleichskriterium nicht aus, auf der anderen gebe es für die Verfügbarkeit, Fehlerhäufigkeit und "design efficiency" noch keine "more sophisticated criteria of comparison". So habe man aufgrund konkreter Erfahrungen viel bessere Kenntnisse über die "physical techniques and design procedures" und verfüge über erste "model of machines", doch "we d'ont even have adequate or recognized evaluation criteria for their assessment and discussion".

Neben Systemstreiten, unterschiedlichen Prioritätensetzungen bei Designkriterien und prinzipiellen Trade-off-Konstellationen beim Computerentwurf kamen bei den vergleichenden Betrachtungen auf den Computerkonferenzen aber auch unterschiedliche Vorgehensweisen im Entwicklungs- und Konstruktionsprozeß zum Vorschein. So bemerkte Jay Forrester (1949, S. 44 f.), daß er beim Whirlwind I das Design mit der CPU und dem Rechenwerk begonnen habe und erst am Ende das "terminal equipment" entwickelt hätte, während es bei den meisten Computerdesignern gerade umgekehrt sei. Der englische Computerentwickler und –hersteller W. S. Elliott (1951, S. 74) erkannte, daß manche Teams in England und USA zuerst das "logical design" der Maschine abschlössen und diese dann nach einem "master plan" entwickelten, während andere inkrementell vorgingen, wobei die Maschine "grew as the ideas came". Er betonte ausdrücklich, daß es sich dabei nicht um nationale Stildifferenzen handele, sondern um individuelle Designmethoden.

Doch neben individuellen Konstruktionsstilen gab es durchaus auch regionale Design-stile. So waren die europäischen Computerentwicklungen, die gegenüber den US-Projekten über viel weniger Finanzmittel verfügten, besonders darauf angelegt, durch raffinierte Algorithmen und Subroutinen mit einer möglichst gering dimensionierten und einfacher strukturierten Hardware auszukommen. Die Tendenz zu minimalistischen Designstrategien war derart ausgeprägt, daß man hier von“ computers with European accents“ sprach (Campbell-Kelly, Aspray 1996, S. 99 ff.; Samuel 1957, S. 14-17). Die meist von Universitäten und kleineren Firmen nach dem „Minimumprinzip“ entwickelten mittleren Anlagen wiesen so in diesen Jahren gegenüber den hoch subventionierten U.S.-Rechnern oft ein deutlich besseres Preis-Leistungs-Verhältnis auf. Nach Zuse waren europäische Rechner auch insgesamt ausgewogener dimensioniert: "Wichtig ist es daher, die Leistung der einzelnen Teile einer Gesamtanlage gut aufeinander abzustimmen. In den USA hatte sich zunächst die Tendenz zu ausgesprochenen großen Geräten durchgesetzt, was zum Teil aber zu erheblichen Fehlinvestitionen und Enttäuschungen geführt hat. Dagegen haben wir in Europa mehr die Tendenz zu mittleren und kleineren geräten, da diese den europäischen Verhältnissen besser gerecht werden." (Zuse, 1952; Zemanek 1958, S. 456 f.; Schuff 1960, S. 3 f.) Aber auch innerhalb der amerikanischen Computer-Landschaft gab es Stilunterschiede. So wurden bei der Debatte über die beste Implementierungsmethode der Logik einer Maschine auf der nationalen Computerkonferenz der USA 1958 zwei Richtungen erkennbar: Die ältere ”Eastern School of Logic Design” mit ihrer Vorliebe für parallele Rechnerstrukturen und Blockdiagramme und die neuere ”Western School” mit ihrer Präferenz für serielle Bauweisen und Boolesche Gleichungen als Modellierungsansatz. Die ältere der beiden Schulen betonte zudem stärker den Kunstcharakter des Computerdesigns, während die neue Richtung mehr auf eine Verwissenschaftlichung setzte (Sprague 1972, S. 690; Langdon 1974, S.98 ff., 158 f.).

Die Wahrnehmung unterschiedlicher Bauweisen und Designmethoden führte sehr bald zu Reflexionen über die Gründe und vor allem auch die Berechtigung solch unterschiedlicher Designschulen und –stile. Einer der ersten, die den Pluralismus architektonischer Bauformen und Designvarianten in Frage stellten, war der Erbauer des EDSAC in Cambridge, Maurice V. Wilkes. Als striktem Anhänger von Neumanns erschien ihm die Entwicklung des Computerdesigns seit dem ENIAC als ein Prozeß der „rationalization, that is the elimination of what is unnecessary“. Die architektonische Vereinfachung bestand in einer generellen Reduktion des „equipment“, vor allem in einer Weglassung von „special de-

vices“. Anstelle einer Vielfalt unterausgelasteter Spezialkomponenten sollten wenige „parts“ möglichst gleichmäßig und in sehr schnellen Prozeßzyklen laufen. Durch die vereinfachte Bauweise sollte auch der Entwicklungsprozeß einfacher und rationeller werden. Sein erklärtes Ziel war „the most efficient way to design an automatic calculating machine“ (Wilkes 1949, S. 150; 1951, S. 182).

Von seiner minimalistischen Philosophie her beunruhigte es ihn, daß das Design von Rechenautomaten zu derart verschiedenen Lösungsvarianten führe: „Two similar groups of engineers with similar backgrounds and assisted by similar groups of mathematicians will, if working independently, produce quite different machines.“ (1951, S.182) Zwar würden die ”general principals of machine design”, die allgemeinste Struktur der ”units“, in vielem übereinstimmen, doch die Arbeitsteilung zwischen den „parts“ würde in jeder ”design group” anders festgelegt. Auch serielle und parallele Operationsprinzipien würden in zahlreichen Varianten vorkommen, wobei die Wahl der Lösung ganz von persönlichen Präferenzen des Designers abhinge (Wilkes 1956, S. 65 f.). Die entscheidenden Ursachen für diese Differenzen sah er vor allen Dingen in speziellen professionellen Faktoren wie unterschiedliche Projekterfahrungen, abweichende Zusammensetzung der Teams sowie in der jeweiligen Ressourcen-Ausstattung und Technologieverfügbarkeit während der Planungsphase. Das Computerdesign beruhte so nach Wilkes sehr wesentlich auf fallweisen Entscheidungen, die auf temporäre und lokale Gegebenheiten reagieren und die nicht selten zu sehr komplizierten und unzuverlässigen Rechnerstrukturen führten.

Die aus ”ad-hoc-manners” und willkürlichen Lösungsentscheidungen resultierende Überkomplexität der Bauweisen wurde bei Wilkes Ausgangspunkt für die Suche nach rationalen Bauformen und rationellen Designmethoden. Daraus entwickelte er seine eigene Architektur- und Designlehre, die er 1951-53 in Grundsatzvorträgen und Aufsätzen und 1956 in dem ersten Lehrbuch über Computer-Design und –Architektur darlegte. Sein ”Best-way“-Designkonzept bestand im Kern aus einem ingenieurmäßigen Elemente-Montage-Modell, das komplexe Strukturen aus einfachen Bausteinen aufbaut. Es ging ihm dabei nicht wie Turing um ein perfektes Design, sondern um eine pragmatische schnell nutzbare Lösung, die mit geringem Personal- und Ressourcen-Einsatz eine hohe Effizienz erzielt. Die Anlagenstruktur sollte durch eine blockartige Verknüpfung möglichst vieler Wiederholteile stark vereinfacht werden, da die Gleichartigkeit von Bauteilen die Komplexität reduziert und so unmittelbar die Verfügbarkeit erhöht. Wilkes sah in diesem Übergang von den höchst

vielfältigen Komponenten zu kombinierbaren standardisierten „small plug-in units“ eine direkte historische Parallele zum Austauschbau in der mechanischen Konstruktion.[27]

Die Übertragung dieser Grundsätze auf die komplexeste Komponente des Rechners, die ”control section”, brachte ihn zur Erfindung des „microprogramming“, bei dem Operationen des Rechners aus ”elementary operations” oder ”micro-operations” zu Mikroprogrammen und diese zu Mikrosubroutinen zusammengesetzt werden. Die Befehlsausführung nach gleichen Prinzipien sollte klare Ablaufstrukturen und Erweiterungsmöglichkeiten schaffen, unüberschaubare ”ad-hoc-alterations to the circuits” vermeiden und so erstmals Ordnung in das Design der Control Unit bringen (Wilkes, Stringer 1953, S. 335-347; Wilkes 1972, S. 973). Auch in der Programmierung suchte Wilkes zusammen mit David Wheeler und Stanley Gill (1951, 1 f.) nach ”Best-way“-Prinzipien, in dem sie das Gesamtprogramm aus möglichst getesteten und erprobten Befehlssequenzen, den elementaren Subroutinen, entwickelte, die ihrerseits zur ”library of subroutines” zusammengestellt und als ”wide utility” von den Programmierern genutzt werden konnten. Auch hier sollte der Rückgriff auf ingenieurmäßiges Vorgehen die willkürlichen Mehrfachentwicklungen zurückdrängen und Komplexität durch modulare Strukturierung beherrschbar machen.

Wilkes’ Design- und Architektur-Konzept war dabei wesentlich von den ihm als ehemaligem Radio- und Radaringenieur bekannten Designprinzipien des ”communication engineering” geprägt, ebenso von seinen frühen praktischen Erfahrungen mit Operations Research-Methoden im Zweiten Weltkrieg (Wilkes 1985, Kap. 4-7). Wilkes übertrug die Elementarisierung von Grundbausteinen und deren Kombination zu austauschbaren Baugruppen, die der englische Computerhersteller Elliott-Brothers und das US-Bureau of Standards mit der Chassisbauweise in die Rechnerschaltkreistechnik eingeführt hatten, nun auch auf auf die Programm- und Architekturebene des Computers (Wilkes 1956, S. 268). Design und Konstruktion eines Rechners waren für ihn dem Wesen nach Ingenieuraufgaben und die aus dem Radar-, Radio- oder Fernsehfach stammenden Ingenieure erschienen ihm deshalb für die ”overall responsibility” des Computer-Design prädestiniert zu sein (ebda., S. 257). Er entwickelte dement-

---

[27] : "This step was not as easy as may appear since the packages had to be designed to rigorous standards so that they would be interchangeable. [...] The advance may be compared with that made in mechanical engineering when interchangeable components were introduced (Wilkes 1972 S. 973)

sprechend das stärker logisch angelegte Architekturkonzept der Von-Neumann-Gruppe zu einer pragmatischen Ingenieurmethodik wieter, die die bewährten Strategien der Komplexitätsreduktion und Designrationalisierung für das ungleich komplexere Feld des digitalen Computers nutzbar machte.[28] Zwar sah auch Wilkes wie von Neumann, Stibitz u. a. den ausgeprägten Zielkonfliktcharakter des Computerdesigns, doch hoffte er der "complexity of logical design" durch das Ingenieurprinzip der "simplification of logic design" und die saubere Trennung von Funktionsbausteinen zu begegnen, mit der die "cross-connections between various units" minimiert und so wechselseitige Abhängigkeiten im Design wie im Betrieb vermieden werden.[29] Die Übertragung der Methoden und Denkweisen des "communication engineering" und des Operations Research hatten Wilkes damit zu einem ähnlich modularen Strukturierungsansatz geführt wie Stibitz mit dem Ebenenmodell eines Computersystems. Beide nahmen im Bereich der Hardware und des Gesamtdesigns in vielem das "Hiding"-Prinzip vorweg, das James Martin, Dijkstra und Parnas Mitte der 60er Jahre für die Software neu entdeckten.

Wilkes Lehrbuch war als eine Zusammenfassung der Erfahrungen der "various design groups" (1956, S. X) der Computer Community in Gestalt einer systematischen Rechnerstruktur- und Design-Lehre angelegt. Obwohl er auf die Elemente-Montage und das One-best-way-Prinzip zielte, war seine Architekturlehre noch nicht von dem historisch realisierten Lösungsraum abgegangen. Seine Systematik wurde demgemäß noch aus einer historisch-deskriptiven Darlegung der Bauformen und einer typisierenden Ex-post-Analyse des eigenen Rechnerdesigns heraus entwickelt. Architekturlehre, Designmethodik, Historie der eigenen Anlage und der „Computer-Zoo" waren noch immer eng verwoben. Wilkes folgte damit dem Beispiel früherer Darstellungen von Hartree, des ERA-Teams und der beiden Booths, die im Rahmen von Gesamtüberblicken über die Computertechnik auch Design- und Architekturfragen behandelt hatten. Das Buch von Douglas Hartree von 1949 "Calculating Instruments and Machines" war aus der Perspektive eines englischen Pioniers von Analogrechnern und Promotors digitaler Computer geschrieben, es war noch ganz als historische Darstellung angelegt und an einzelnen Maschinen

---

[28] Für anthropromorphe oder biokybernetische Metaphern war daher in seinen Architekturreflexionen kein Platz, die Organ- und Neuronennetz-Vorstellungen wurden hier konsequent von dem Komponentendenken und Systembeziehungen der Elektronik abgelöst (vgl. Wilkes 1968, S. 6)

[29] Wilkes definierte die Komplexität durch das Ausmaß nicht-logikgemäßer Querverbindungen in einem Rechner: "By the complexity of a machine I mean the extent to which crossconnections between various units obscure their logical interrelation. " (1951, S. 182)

orientiert. Die folgenden Bücher hatten jeweils noch längere historische Abschnitte und gingen ansonsten bereits systematisch vor, wobei sie jedoch noch die eigene Entwicklung als Referenzmaschine der Erörterung und der Vergleiche zugrunde legten.

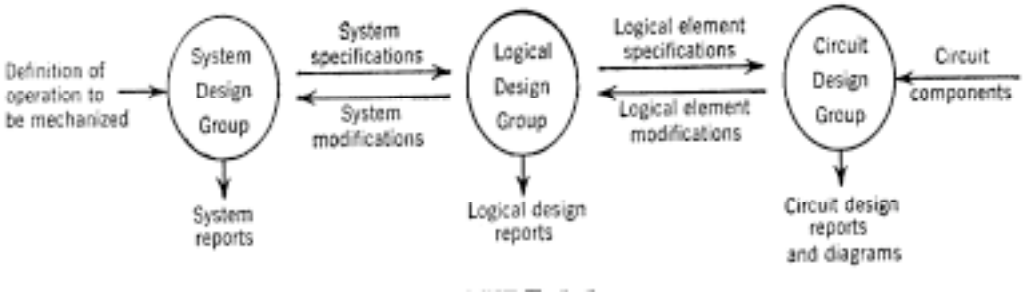
Der aus internen Berichten der "Engineering Research Associates" hervorgegangene State-of-the-Art Report "High-Speed-Computing Devices" von 1950 war eine systematische Einführung in das Gesamtgebiet analoger und digitaler Rechenautomaten, Lochkartenmaschinen und in die jeweiligen Rechenverfahren und Anwendungen, das zeitweise den Rang einer Enzyklopädie hatte (Hoffmann, 1962, S. 679). Die darin enthaltenen Designbetrachtungen folgten, typisch für ein ehemaliges "war-time team", einem einfachen Designphasenmodell und Engineering-Konzept. Sie lehnten sich im übrigen bei der idealtypischen Verallgemeinerung der ERA-Bauform eng an die von Neumann-Gruppe an. Andrew und Kathleen Booth stützten sich in ihrem Buch "Automatic Digital Calculators" von 1953 sehr weitgehend auf den von ihnen am Birkbeck-College in London entwickelten ATE(X)C-Computertyp, einer besonders kleindimensionierten von Neumann-Maschine. Bei der Darstellung der Prinzipien des "overall design" und der "organization" der einzelnen "units" verwiesen sie aber auch immer wieder auf Lösungen anderer Computer. Nachfolgende Lehrbücher lösten sich zunehmend von der historischen Vergleichsbasis realisierter Bauwiesen und von dem unmittelbaren Erfahrungshintergrund spezieller Anlagen.

Gefördert wurde dieser Übergang zu systematischeren Darstellungsformen durch die Rezeption systemtechnischer Methoden, die seit der Mitte der 50er Jahre sehr schnell zu einem Kernbestand der Allgemeinen Ingenieurmethodik aufgerückt waren. Operations Research-Methoden und vor allem die komplexere Methodik des „Systems-Engineering", die seit ca. 1943 in den Bell-Laboratorien zur Steuerung großer nachrichtentechnischer Entwicklungsvorhaben entstanden war, begannen sich in der zweiten Hälfte der fünfziger Jahre auch an Hoch- und Fachschulen als neues Forschungs- und Ausbildungsgebiet zu etablieren (Hall 1989, S. 10-24). Das erste Lehrbuch über „Systems Engineering", das 1957 von Harry H. Goode und Robert E. Machol vor allem auf der Basis von Erfahrungen bei Bell-Lab und AT&T verfaßt worden war, wurde bald ein wichtiges Methodenvorbild für die Computer Community, bot es doch ein idealtypisches Phasenmodell des „Systems Design" für Computer- und Automatisierungsvorhaben.

Die erste voll vom System Engineering geprägte Architekturlehre war das 1958 erschienene Lehrbuch "Logical Design of Digital Computers" von Montgomery Phister, Jr. Als Leiter der Entwicklung und Vizepräsident der „Scientific Data Systems" hielt er bereits seit Mitte der 50er Jahre Abendkurse über "digital computer design" an der Universität von Los Angeles ab. Unter dem Einfluß des "System Design" verschob sich bei ihm der Schwerpunkt von der bisher dominierenden Organstruktur- und Bauformenlehre zur Organisationslehre des Designprozesses mit der Modularisierung des Konstruktionsgegenstandes und der Phasenunterteilung des Entwicklungsprozesses als Kernmethoden. Das Computerdesign wurde hier von vornherein als rationelle Komponentenauswahl und –verknüpfung angesehen mit dem Ziel, "the most effective configuration of those components" zu finden (S. 2). Phister sieht dabei die komplexen Beziehungen zwischen den Rechner-bestandteilen nicht mehr wie von Neumann als eine interdependente Organ- und Problemstruktur, die eine sequentielle Bearbeitung ausschließt, sondern für ihn löst sich mit Hilfe der systemtechnischen Dekomposition das "total design problem" in eine arbeitsteilungsgerechte Komponentenstruktur auf: "the design of a digital computer is usually carried out by breaking the computer down into a number of different functional parts each of which may be designed *independently* from others" (S. 340, meine Hervorhebung). Das Design erhält so den Charakter eines arbeitsteiligen Herstellungsprozesses, der entsprechend als Produktlinie modelliert werden kann. Damit wird für Phister das Phasenmodell zum zentralen Organisationsmittel des Designs und zur Leitlinie seiner Architekturlehre.

Das Gesamtdesign wird bei ihm in die drei "basic steps": system design, logical design und circuit design aufgeteilt, die von drei einander gleichgestellten „design groups" getragen werden. Die Hauptphasen werden dann weiter in einzelne Designschritte von der Systemanalyse bis zum Test und zur Inbetriebnahme unterteilt. Daß diese saubere Phasenbildung das Komplexitätsproblem des Designs nicht löst, sondern nur wegdefiniert, wird bei der Frage der Lösungssynthese und Teileintegration deutlich. Denn hierbei kommt das systemtechnische Modularisierungskonzept nicht ohne eine enge Kooperation der drei Designergruppen aus. Ein ständiger Informationsfluß über Systemspezifikationen und vor allem System- bzw. Logik-Modifikationen ist nötig, um die Probleminterdependenzen des Designs in den Griff zu bekommen. Die Synthese gelingt auch nur, wenn die Mitglieder jeder der drei Gruppen die Probleme und Schwierigkeiten der anderen gut kennen. Entgegen der wohldefinierten Aufteilung des Konstruktionsgegenstandes herrscht am

Ende eine „somewhat complex relationship between these three groups“ (S. 3). Das gilt vor allem für den Entwurf der „control unit“, der zum Abschluß des Designs" das „total design problem" noch einmal in voller Schärfe auftreten läßt. Da dieses nicht durch einfaches Zusammensetzen der Teile zu lösen ist, schlägt Phister eine Art Prototyping mit sehr einfachen Computermodellen vor, die dann sukzessive zu komplexeren Anlagen skaliert werden. Dabei kann unter Umständen die Revisionsphase zu einem Rearrangement oder gar einem "completely new design of the whole computer" führen (S. 340 f.). Ein solch iteratives Vorgehen mußte aber schnell an Kostengrenzen stoßen und widersprach zudem Erkenntnissen von Computerdesignern wie Werner Buchholz, wonach "small pilot models" im Computerbereich im Unterschied zu anderen Techniken nicht allzu nützlich waren, denn "they do not exhibit many of the important characteristics of the full-scale machine" (1953, S. 1262).



Nicht-hierarchische Kooperation der drei zentralen Designgruppen bei Phister (1958, S. 3)

Während bei Phister das ungelöste Komplexitäts- und Syntheseproblem des Designers hinter der klaren systemtechnischen Modellierung immer wieder durchscheint, geht dieses Problembewußtsein in folgenden System Engineering-Konzepten des Computerdesigns vielfach verloren. So stellten sich in den bereits für die reguläre Hochschulausbildung verfaßten Lehrbüchern über Computerdesign von Ivan Flores von 1960 und 1967 Entwicklung und Konstruktion eines Rechners als relativ problemlose Zusammenfügung von "elementary logical elements" zu "operational" bzw. "functional units" und weiter zu "large functional units" dar (Flores 1960, S. VI). Die Systemkomposition folgt dabei der hierarchisch aufgebauten "chain of synthesis": "The system is built from subsystems which in turn are built from smaller building blocks. This is like boxes within boxes" (Flores, 1967, S. 6). Auch die Programmierung wurde nun konsequent als hierarchische Komponentenstruktur begriffen: Ganz nach dem

Vorbild der Chemie steigt bei Flores die ″hierarchy of programming″ vom Atom ″command″ über die Moleküle ″loop″ ″subroutine″ und ″routine″ zu den komplexesten Verbindungen der ″library of programs″ auf.

Die Methode der System-(De)Komposition förderte so die Bildung von Modulhierarchien und Ebenenmodellen, die den Entwicklungsgegenstand arbeitsteilungsgerecht modellierten und so die Grundlage für elaborierte Prozeßketten werden konnten. Das Problem des systemtechnischen Design- und Architekturansatzes war jedoch die Inhaltsleere des Systembegriffs und der Schematismus der Strukturierung. Es nützte dem Designer wenig, den Computer als ein System im Sinne einer ″aggregation of parts″ bzw. einer ″configuration″ von Subsystemen oder als eine ″machine with inputs and outputs″ zu sehen. Das Komplexitäts- und Syntheseproblem wurde damit nicht gelöst. Nur in Teilbereichen wie den Schaltkreisen, Speichern und Rechenwerken waren orthogonale Strukturen und funktionale Modularisierungen und damit Produktlinien möglich, nicht aber in der Kontrolleinheit und dem Gsamtdesign, wo es komplizierte kommunikative Beziehungen und Wechselwirkungen zwischen den Funktionseinheiten gibt, die sich einer klaren theoretischen Beschreibung und damit einer "systematischen Behandlung" entziehen (Ganzhorn, 1978, S. 20). Durch die ″system composition″ wurde zwar aus der komplexen Problemstruktur eines Computers eine überschaubare Struktur von Aufgabenpaketen, doch damit wurden auch die Interdependenzen der Designmerkmale, alle Zielkonflikte und Entscheidungsalternativen der verschiedenen Bauformen ausgeblendet oder zu ″restrictions″ bzw. ″limiting factors″ marginalisiert (Flores 1960, S. 32). Man ging davon aus, daß die Designkriterien bei der anfänglichen Anforderungsanalyse durch die System-Designer-Gruppe komplett ermittelt und in eindeutigen Spezifikationen festgelegt werden und daß im übrigen durchgängige Designfragen und –konflikte zwischen den noch weitgehend gleichgestellten Teams der System-, Logik- und Schaltkreisdesigner durch informelle Kooperation abgestimmt werden.

Durch Annäherung an die Modul- und Synthesetechnik der Elektro- bzw. Elektronikkonstruktion sowie an systemtechnische Methoden der System-, Produkt- und Prozeßmodellierung schien das Computerdesign weitgehend zu einer lehr- und lernbaren Konstruktionstechnik zu werden, die aus vorhandenen Bausätzen und Lösungsmustern anforderungsgemäße Anlagen konfiguriert und montiert. Doch entsprach diese an etablierten Technikzweigen orientierte Entwicklungs- und Konstruktionsmethodik noch nicht dem damaligen Stand der Computertechnik. Trotz der zunehmenden Durchsetzung der ″klassischen

Von-Neumann-Architektur" war die Entwicklung noch nicht so weit abgeschlossen, waren die Bauformen noch nicht so standardisiert und die Designmethoden noch nicht so ausgereift, daß ein am Austauschbau und Fertigungsprozessen ausgerichtetes Elemente-Montage-Konzept das adäquate Muster für die Technikwissenschaft und Designlehre des Computers darstellte. Für John Mauchly war deshalb 1963 das "digital computer engineering" noch immer "an art to be learned": "Too often the subject has been presented as a set of immutable principles which are directly and inexorably applied to satisfy predefined specifications. Rarely has the beginner been permitted a glimpse of computer design as a back-and-forth play involving artistic creation using imperfectly understood components" (Mauchly 1963).

Engineeringmethoden und System Engineering durchdrangen in der zweiten Hälfte der 50er und ersten Hälfte der 60er Jahre Designlehren und "Theorien des Computers". Es erschienen nun auch Lehrbücher und Monographien zum "computer engineering", so von Bartee (1960), Ledley (1960), Bartee/ Lebow/ Reed (1962) und Gray (1963). In diesen Lehrbüchern wie auch in den Einführungen in „Digital Computer Principles „ bzw. „Fundamentals" von Irwin, Chu und Hintze erschienen Grundsatzfragen der Rechnerarchitektur als entschieden und gelöst, so daß sich die Rechnerentwicklung hier als ein relativ problemloses Auswählen und Arrangieren von logischen, elektrischen Schaltkreisen und Komponenten darstellte: „The computer configuration is formulated by choosing the registers, matrices, memory, and input-output devices." (Chu 1962, S. 307) Es überwiegt dabei die Bottom-up-Darstellung, bei der, ausgehend von den Zahlensystemen, stufenweise aus „building blocks in an engineering sense" die Schaltkreise, die Komponenten und schließlich das Gesamtsystem entsteht, wobei dieses oft in Form einer idealisierten Minimalkonfiguration vorgeführt wird (Gschwind 1967, S. III). Eine Reihe von Handbüchern stellt dagegen den Computerentwurf von einem „system-design viewpoint" als einen Top-down-Prozeß dar, - Alan Perlis sprach direkt von „top-down-teaching" – in dem alle Designvorgänge strikt durch das systemtechnische Phasenmodell und Bausätze bzw. Baukästen auf jedem „level of the computing hierarchy" geordnet werden (Bartee, Lebow, Reed 1962, S. 9, 205; Chu, 1962; Gray 1963; vgl. Bauer, Goos 1970, Vorwort).

Doch gelegentlich tauchen auch in den ingenieurwissenschaftlichen Architekturkonzepten Zweifel an einem allzu schematischen Ablauf auf und geraten Lösungsvielfalt und Designkonflikte in den Blick. So etwa bei Harry J. Gray (1963, S. 37) : „Because of the branching, which is characteristic of synthesis,

there is no unique solution unless some criterion, such as the previously defined efficiency criterion, is used to select one from the many possible designs.“ Und Hans W. Gschwind (1967, S. 160) gelangt im Laufe seiner Darstellung zu dem Resultat, daß die Rechnerentwicklung neben den Building-Block-Konzepten einer besonderen Erörterung der „design philosophy“ bedürfe, denn: „the organization of digital computers is still in a state of evolution. We do not yet have the ideal computer (if there is an ideal computer at all). Designers continously conceive ideas which change the basic structure of computers in one respect or another.“ Die zahlreichen noch ungelösten Komplexitäts-, Synthese- und Kooperationsprobleme, die durch die Größen- und Leistungssteigerungen der Anlagen ständig zunahmen, sowie die Offenheit der Bauwiesen und die Vielfalt der Nutzeranforderungen waren so letztlich der Grund dafür, daß sich aus diesen ingenieurwissenschaftlichen Denkweisen keine Disziplin ”Computer Engineering” nach dem Vorbild des Electrical oder Communication Engineering entwickelte. Im Unterschied zum Software Engineering wurde nämlich in der Computerdesign- und –konstruktionslehre gerade nicht das *Engineering* zum namengebenden Leitbild, sondern die *Architektur*.

Design-orientierte Architekturkonzepte

Hierarchisches Architekturkonzept: Speiser 1961

- Hierarchie der Organsationsstufen der Anlage und des Entwurfs, Systemaufbau einer Anlage als kunstvolle Integration heterogener, widersprüchlicher Forderungen, Netzvorstellung der Bedingungen und Beziehungen innerhalb eines Rechners,Übergang zu systemtechnischen, engineering-orientierten Architekturkonzepten

Einführung des Architekturbegriffes: Brooks 1961/62

- Definition der Rechnerarchitektur von der Integrations-funktion der Chefdesigner her: Abstimmung von Nutzerinteressen mit Rechnersystemeigenschaften
- Herausfilterung von Architekturmerkmalen, Gütekriterien und Designprinzipien aus Spitzenprodukten und dem Erfahrungsschatz der Designerelite

Systemfamilien-Architekturkonzept der IBM /360 Amdahl, Blaauw, Brooks, 1964

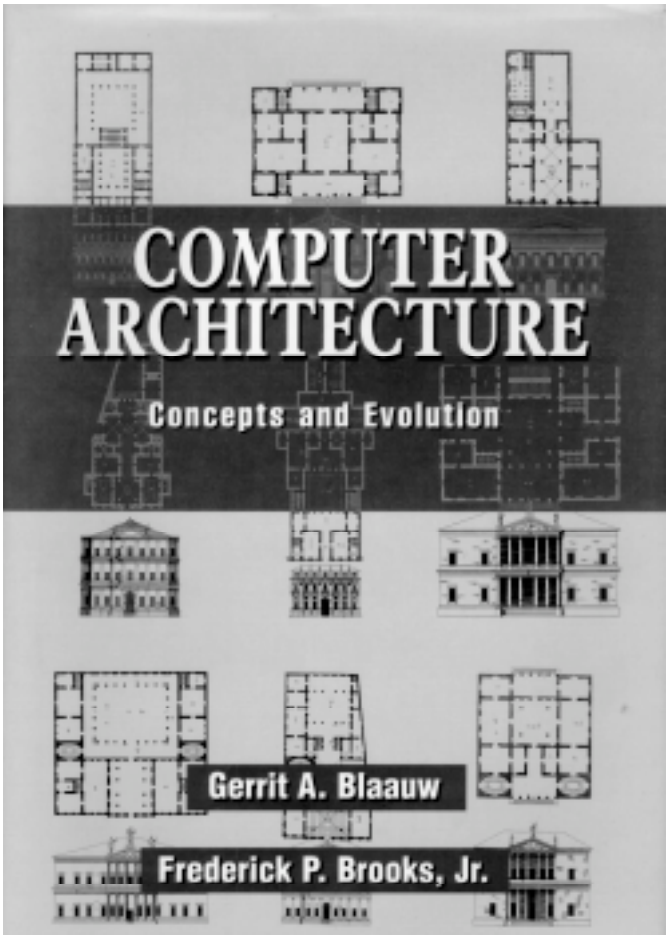
- Unternehmensstrategische Ausrichtung des Architekturkonzeptes
- Fokussierung auf Kompatibilität und auf das einheitliche Erscheinungsbild gegenüber den Programmierern
- Lösung der Architektur von der Implementierung (innere Rechnerstruktur) und Realisierung (physikalische Umsetzung)

Ästhetisch-funktionales Interfacekonzept der Architektur: Blaauw, Brooks, 1972/75/97, Hayes, 1978, Lipovski, Doty, 1978, Dennis, Fuller, Ackerman, 1979, Garside, 1980

- Begründung der Architektur auf der Grundlage von Qualitäts- bzw. Gütekriterien und eines Katalogs von Designprinzipien für das gesamte Benutzer-Interface ('Fassadensicht' der Architektur)
- Qualitätsorientierter, intentionaler Architekturbegriff, aristokratisches Designkonzept der Computerarchitektur mit dem Schwerpunkt der funktionsgemäßen Nutzungs-qualitäten

Desigkonfliktmodelle von Computerarchitekturen: Speiser, 1961/64, Yourdon, 1972, Abrams, Stein, 1973, Thurber, Masson, 1979

- Trade-off-, Constraints-, und Bottleneck-zentrierte Architekturkonzepte



4. Die Professionalisierung der "computer architects":
Die Chefdesigner und das hierarchische Komplexitäts-Management

Die ingenieurwissenschaftlichen und systemtechnischen Dekompositionsstrategien hatten zwar besonders mit der Modularisierung und hierarchischen Ebenenbildung Ansatzpunkte zu einer arbeitsteiligen Bewältigung der Designkomplexität geliefert, dabei aber wesentliche Designentscheidungen, Synthese- und Organisationsprobleme ausgeklammert. Anfang der 60er Jahre entstand daher ein Architekturkonzept, das die Betrachtung gerade auf die bisher unterbelichteten Syntheseaspekte und technisch-organisatorischen Zusammenhänge des Designs und der Struktur eines Computersystems fokussierte. Die Perspektive verschob sich hier vom Best-way-Prinzip, von der Elementarisierung und Bausatz-Montage zur Organisation der Gesamtstruktur und zum erfahrungsbasierten Management der Abstimmungs-, Integrations- und Syntheseprozesse hin. Nicht mehr die möglichst rationelle Entwicklung und Konstruktion war das vorrangige Ziel, sondern die nutzer- bzw. nutzungsbezogene Produktoptimie-

rung. Als Leitbegriff für diesen Perspektivwechsel und erweiterten Ansatz der Komplexitätsbewältigung im Rechnerdesign wurde um 1960 nun der Architekturbegriff in die Computer Community eingeführt.

Es hatte seit den Traktaten der Pionierphase eine ganze Reihe von Vorformen des Begriffs der Rechnerarchitektur gegeben, um den übergeordneten Charakter von Struktur- und Designentscheidungen zu kennzeichnen. So sprach man beim ENIAC von der ″description of the machine as a whole″ bzw. vom "arrangement of the computing system″, beim UNIVAC vom ″over-all picture of the UNIVAC-design″, beim Whirlwind I vom ″system layout″ sowie von den "general principles of design" der IBM 701.[30] Die meist verwendeten Bezeichnungen, vor allem in den Traktaten und frühen Lehrbüchern, waren aber ″organization″ bzw. ″configuration" und das von John von Neumann protegierte ″logical design″, das damals noch, wie Wilkes sich erinnert, die Bedeutung des späteren Architekturbegriffes hatte. So machte auch Alan L. Leiner von der Computerabteilung des ″National Bureau of Standards″ in einer Skizze einer ″basic theory of design″ auf der internationalen IEE-Konferenz ″Convention on Digital-Computer-Techniques″ von 1956 ″logical design″ zum Kernbegriff und definierte ihn wie folgt: ″[...] 'logical design' may be defined as the process of figuring out how to organize electronic devices of known logical properties into an integrated system that can perform specified tasks of computation and control" (Leiner, 1956, S. 123). Doch trotz der Ausrichtung auf eine übergeordnete, prinzipielle Perspektive setzten sich nicht diese analytischen, logischen oder systemtechnischen Bezeichnungen durch, sondern gerade der ganzheitlich-gestalterische Architekturbegriff.

Die Einführung der Architektur-Metapher in den Computerbereich ist deshalb außergewöhnlich, weil es außerhalb des Bausektors so gut wie keine Technikwissenschaft gibt, die zentrale Forschungsgebiete oder Teildisziplinen damit belegt. Denn nach der Verwendung in frühneuzeitlichen Techniktraktaten und Kunstlehren wie in Joseph Furttenbachs ″Architectura civilis″, ″martialis" und ″navalis″ Ende des 16. bzw. Anfang des 17. Jahrhunderts hat der Architekturbegriff in den Technikwissenschaften kaum noch eine Rolle gespielt. Es finden sich jedenfalls nur sehr wenige Beispiele in der Netzwerkgestaltung und ansonsten nur gelegentlicher unspezifischer Gebrauch. Mit der Wahl des Architekturbegriffes stützte sich die Computer Community also nicht mehr auf die

[30] H. H. u. A. Goldstine 1946, S. 10; Eckert, Weiner 1951, S. 175; Everett 1951, S. 139; Buchholz 1953; zum folgenden siehe Hartree 1949, S. 12; A. u. K. Booth 1953, S. 22; Bowden 1953, S. 67 ff.; Wilkes 1985, S. 147.

Mechanik-, Elektro- und Elektronikkonstruktion, die Ingenieurwissenschaften oder auf Mathematik und Naturwissenschaften, aus denen sie bis dahin überwiegend ihre Strukturierungs- bzw. Darstellungsmodelle und Konzeptmetaphern entlehnt hatte. Sie trat nun - in voller Übereinstimmung mit den wissenschaftssoziologischen Entwicklungsmodellen - selber begriffsschöp-fend und modellbildend in Erscheinung. Der Rekurs auf die Architekturmetapher war dabei keineswegs ein Zufall, die Rekonstruktion der Begriffsgeschichte zeigt vielmehr, daß die Begriffswahl von Beginn an in einem spezi-fischen professionellen und sozialen Entstehungskontext stand: Sie war Ausdruck designmethodischer und arbeitsorganisatorischer Veränderungen, wie sie vor allem bei der großskaligen Computerentwicklung kommerzieller Hersteller in der zweiten Hälfte der 50er Jahre auftraten. Während das Engineering-Konzept der Rechnerarchitektur vor allem im Hochschulbereich und Entwicklerkreisen mit "war-time team"-Hintergrund angesiedelt war, entstand das Design- und Management-orientierte Architekturkonzept in der Designerelite der IBM-Forschungs- und Entwicklungslaboratorien, bezeichnenderweise im Rahmen der beiden ambitioniertesten Computerentwicklungen der Zeit, des Stretch-Projektes und der 360er Systemfamilie von IBM. Doch die erste ausführliche Formulierung des organisatorischen Architekturansatzes findet sich noch vor dem öffentlichen Erscheinen des Begriffes bei dem Leiter eines der europäischen IBM-Forschungslaboratorien, bei Ambros Speiser in Zürich.

Speiser hatte in den Jahren 1947-49 bei Forschungsreisen die Anlagen von Wilkes, Aiken, Stibitz und von Neumann kennengelernt und aus einer vergleichenden Perspektive zusammen mit Heinz Rutishauser und Eduard Stiefel 1950 den ersten deutschsprachigen Gesamtüberblick über den Stand der Technik verfaßt. Im gleichen Jahr legte Speiser einen Designleitfaden vor, in dem er einen kleindimensionierten, auf "Materialersparnis" bedachten Hochschulrechner als bewußt anwendungsbezogene Strukturentscheidung und Lösungskombination explizierte, wobei er gezielt Architekturkomponenten der Anlagen von v. Neumann, Aiken, Wilkes und Zuses Z 4 verknüpfte. Speiser betrachtete dabei, noch ganz in der Von-Neumann-Tradition, den Rechner als ein Gleichgewichtsmodell interdependenter Organe und zugleich als ein Designkonfliktmodell, bei dem gegenläufige Ziele austariert und ein "vernünftiges Gleichgewicht" zwischen Hardware- und Programmieraufwand hergestellt werden müßten (Speiser 1950, S. 9 f., 16). Wie von Neumann erkannte auch Speiser das sich sich aus dem Interdependez-Problem des Designs ergebende Darstellungsproblem: "Diese Bausteine sind funktionell aufs engste miteinander verknüpft, und eine Verbesserung im einen Teil muß

notwendigerweise von einer entsprechenden Anpassung in allen übrigen Gliedern begleitet sein, damit sie wirklich zur Auswirkung gelangt. Dabei ist, wenn im folgenden die verschiedenen Hauptteile getrennt behandelt werden, doch in Erinnerung zu behalten, daß die Fortschritte in jedem Fall eng miteinander verknüpft sind." (Speiser, 1962, S. 70) Durch seine Erfahrungen im IBM-Entwicklungsmanagement gewann er jedoch die organisatorische Perspektive und Einblick in hierarchische Lösungs-muster für das Komplexitätsproblem, so daß er 1961 in seinem bei Springer erschienen Lehrbuch "Digitale Rechenanlagen" das erste elaborierte hierarchische Architekturkonzept vorlegen konnte.[31]

Der Aufbau des Rechners aus Elementen mit gegebenen elektrischen Eigenschaften ist danach keine primär technische oder physikalische Aufgabe, bei der es auf Naturgesetze ankomme. Er ist vor allem "organisatorischer Natur", da die Anforderungen der "Benützer" in eine entsprechende Systemstruktur und Technologie umgesetzt werden müssen: "Nun sind aber die beiden Fragenkreise unzertrennlich miteinander verbunden, und man kann nicht etwa so vorgehen, daß man zuerst alle technischen Aufgaben abschließend löst und sich dann den organisatorischen zuwendet, oder umgekehrt. Eine Anlage besitzt verschiedene Organisationsstufen; sie beginnen mit den einzelnen Zellen und rücken bis zur Gesamtkonzeption der Maschine vor. Auf jeder dieser Stufen sind organisatorische und technische Forderungen, die sich oft widersprechen, in Einklang zu bringen". (Speiser 1961a, S. 35) Das Rechnerdesign wie die entsprechende Entwurfslehre erhalten auf diese Weise die Form einer aufsteigenden "Hierarchie" von "Organisationsstufen". Die Anlagenstruktur und das Designprozeßmodell werden so gleichermaßen hierarchisch durchstrukturiert und das Architekturkonzept dadurch soziomorph. Doch obwohl Speiser damit in vielem das Schichtenmodell der Rechnerarchitektur vorwegnahm, hielt er bei den Bezeichnungen an den älteren Begriffen und Metaphern fest: er sprach von "Gesamt-" oder "Systemorganisation", von "systemmäßigen Aufbau" sowie von der "Anatomie" und der "Physiologie" der Rechenanlage und ihrer "Organe" (vgl. auch Speiser 1961b, S. 76 ff.).

Erst ein Jahr nach Speisers Hierarchiemodell des Rechnerdesigns brachte Herbert A. Simon die hierarchische Strukturierung von Systemen mit dem Architekturbegriff in Verbindung, allerdings nahezu ausschließlich in der Überschrift seines berühmt gewordenen Aufsatzes "Architecture of Complex-

[31] Nach persönlicher Mitteilung von Ambros Speiser beruhte sein im Buch entwickeltes Architekturkonzept auf seinen IBM-Erfahrungen.

ity". Simon (1962 S. 63 f., 71, 76) verglich darin Ordnungsprinzipien mechanischer, physikalischer, biologischer und sozialer Systeme. Hierarchiemodelle, die schon seit langem Gegenstand von politischen Theorien den Staatswissenschaften und vor allem der Soziologie waren, waren bereits in der ersten Hälfte der fünfziger Jahre durch Ludwig von Bertalanffy über die Systemtheorie in die natur- und ingenieurwissenschaften eingeführt worden. Seit Beginn der sechziger Jahre tauchten in vielen Bereichen der Technik, Naturwissenschaften und Sozialwissenschaften Hierarchiekonzepte als ideale Strukturierungsansätze für komplexe Strukturen und Abläufe auf. Simon wollte mit seiner "theory of hierarchy" analog zur Kybernetik eine "nicht-triviale Theorie" komplexer Systeme schaffen und das Hierarchiekonzept dabei von der Systemtheorie lösen. Besonders mit seinem bekannten Uhrmacher-Vergleich demonstrierte er, wie hierarchische Strukturen die Probleme arbeitsteiliger Prozesse besser lösen als die Selbstorganisationsprinzipien der Algemeinen Systemtheorie. Seine Kernaussage war, daß sich alle natürlichen und artifiziellen Systeme jeweils von einfachen zu komplexen Strukturen entwickeln und dabei schon von sich aus zu hierarchischen Strukturbeziehungen tendieren.

Die Hierarchie bzw. Architektur als Strukturgestalt komplexer Systeme war für Simon aber zunächst nur eine Metapher bzw. Analogie, die erhellend sein kann, aber auch in die Irre führen könne. So warnte er insbesondere vor einer Überschätzung der "hierarchic formal organization" mit eindeutiger Subordination. Diese seien weder die normale noch besonders effiziente Organisationsform, die Regel in sozialen wie nicht-sozialen Systemen seien eher Hierarchien ohne Unterordnung der Subsysteme: "In fact, even in human organizations, the formal hierarchy exists only on paper; the real flesh-and-blood organization has many inter-part relations other than the lines of formal authority" (ebda., S. 64). Da die "description of complexity" auf der Ebene allgemeiner Organisationsstrukturen angesiedelt ist, geht Simon hier mit keinem Wort auf die konkrete Bewältigung der Komplexität von Hard-und Softwaresystemen ein. Erst 1973 legte Simon eine hierarchie-theoretische Analyse komplexer Rechner- und Programmsysteme vor, die alle Vorteile hierarchischer Modul- und Ebenen-Strukturierung zusammenfaßt. Zugleich wies er aber mit seiner „theory of near-decomposabi-lity“ darauf hin, daß auch in wohl strukturierten hierarchischen Gebilden zwischen den Ebenen, Modulen und Komponenten Wechselbeziehungen bestünden, die keine starren Regelungen, sondern lose Kopplungen erforderten. Allerdings spricht Simon hier nicht mehr von Architektur, sondern von einer „General Theory of Complexity“ (Simon 1973). Obwohl Simons Beiträge von großer Bedeutung für das Rechner- und

Programm-Design waren, wurden sie nicht, wie immer wieder behauptet, der eigentliche Ausgangspunkt des Architektur-Konzeptes in der Informatik. Es blieb vielmehr Frederick Brooks vorbehalten, den Architekturbegriff endgültig mit dem Computer-design zusammenzubringen und ihn in dieser Bedeutung als erster im Jahre 1962 in die Fachöffent-lichkeit der Computer Community einzuführen.

Als einer der Chefdesigner im Stretch-Projekt und als Projektleiter der 360er Systemfamilie verwendete Brooks den Architekturbegriff in internen Memoranden der IBM sogar schon seit 1959. Brooks, ein Schüler Howard Aikens, war im Rahmen seiner Harvard-Dissertation von 1956 auf das Problem der Zielgruppen-spezifischen Gestaltung von Computersystemen gestoßen, denn in ihr hatte er systematisch untersucht, welche Änderungen im Rechnerdesign und in der Entwurfsmethode sich beim Wechsel von wissenschaftlichen zu kommerziellen Nutzerkreisen ergeben. Ähnliche Überlegungen im Stretch-Projekt bewogen ihn dann, alle nutzerbezogenen Aspekte des Designs mit dem Architekturbegriff von dem Wirrwarr der Detailentscheidungen abzugrenzen. Dabei ging der erste Anstoß zur Begriffsbildung offenbar von dem Stretch-Mitarbeiter Lyle R. Johnson aus, der in dem vertraulichen IBM-Report RC-160 "A Description of Stretch" vom 10.11.1959 mit "system architectonics" bzw. "architecture" die über das "circuit" und "logic design" hinausgehenden Designaufgaben gekennzeichnet hatte (Johnson, 2000, S. 70 f.) Mit dem ”User-” und Nutzungs-bezogenen Gebrauchswert des Designs gab Brooks dem Begriff jedoch eine neue Ausrichtung, die auch in der Folgezeit der Kern seines persönlichen Architekturkonzeptes blieb. Dessen erste systematische Darstellung erfolgte in dem 1961 von dem IBM-Systemdesigner Werner Buchholz herausgegebenen und 1962 erschienenen Buch ”Planning a Computer System: The Project Stretch”. In ihm wurde die Designphilosophie und die Funktions- und Strukturmerkmale eines der größten bis dahin gebauten Rechner dargestellt, des Stretch-Computers bzw. des IBM 7030. Obwohl dieses System am Ende kommerziell wenig erfolgreich war, wurde es durch seinen radikalen Bruch mit traditionellen Designmethoden und Bauweisen ungeplant zum Erprobungsfeld für die spätere IBM-360-Systemfamilie und darüber hinaus zum ”exciting center for computer-organization studies” und war damit für die Entstehung des Architekturbegriffes geradezu prädestiniert. (vgl. Bashe u.a. 1986, S. 443).

Zentrale Designziele dieses in den Jahren 1954 -61 entstandenen Großrechners war die Verhundertfachung der Leistung des bisherigen Paradepferdes der IBM-

Mainframes, des IBM 704, und die Zusammenführung der bisher getrennt verlaufenen Entwicklungen wissenschaftlicher bzw. militärischer und kommerzieller Rechner. Hierfür wurde eine Integration der unterschiedlichen Designmerkmale und damit eine Überwindung bisheriger Systemstreite erforderlich, so vor allem bei der binären und dekadischen Zahlendarstellung, der festen und variablen Wortlänge sowie bei der Fließ- und Festkomma-Arithmetik.[32] Aus dem Integrationsbemühen ergab sich eine grundlegend neue "logical organization of a computer": Die Ausweitung und interne Strukturierung der Funktionalität, die Auslösung der Funktionen durch ein umfassenderes und zugleich effizienteres "instruction set" und eine effektivere Aufteilung der Funktionen auf die Hardware- und Softwareressourcen des Systems mit dem Ziel einer weitgehenden Überlappung bzw. Parallelisierung von Prozeduren (Brooks, Blaauw, Buchholz 1960, S. 376). Dabei wurde der Instruktionsatz ohne Rücksicht auf Engineering-Probleme von den Programmieranforderungen her definiert, während er sonst meist umgekehrt von Anlagenbauern ohne vorherige Kooperation mit den Programmierern entwickelt wurde (Lewin 1980, S. 56). Die zentralen Designziele wiesen hierbei eine teilweise Gegenläufigkeit, ja sogar Widersprüchlichkeit auf: "general purpose"-Orientierung auf hohem Niveau und zugleich Anpassungfähigkeit an unterschiedliche Nutzerkreise; eine starke Ausweitung der Funktionalität bei einer Erhöhung des Bedienungskomforts; die Vereinfachung der Herstellung und alles bei einer bislang nicht erreichten Anhebung der Leistungsparameter. All dies zwang das Designerteam des Stretch zu besonders systematischen Überlegungen über den Zusammenhang von Nutzungsanforderungen der "user", übergeordneten Designmerkmalen der internen Organisation der Maschine, der Allokation der Funktionen im Ressourcenspektrum sowie der Art und Weise des Umgangs der Benutzer mit dem Rechner. Das "engineering design" wurde so in dem kleinen Entwicklerteam selbst zum Gegenstand intensiver "design considerations", deren Ergebnis man sogar nach außen hin in Buchform dokumentierte. Der Titel des Bandes, "Planning a Computer System" unterstreicht den Anspruch der Autoren, mit dem Erfahrungsbericht über die "design phase of a new generation computer" der Designer-Community zugleich ein Vorgehensmuster für die künftige Computerentwicklung zu geben. [33]

---

[32] Brooks 1965, S. 89; Hurd 1981, S. 179; zur Stretch-Entwicklung vgl. bes. Bloch 1959; Speiser 1961b; Bashe u.a. 1986, S. 416-458 und Buchholz 1962a, passim.

[33] Vgl. Bengt Carlson im Vorwort: "This book will be invaluable as a guide and reference source for computer development in the future," (Buchholz, 1962, S. VI)

Architektur-Begriff bei Frederick Brooks

- ***Prozeßcharakter*** **in Gestalt einer kunstvollen Designmethodik, die Nutzeranforderungen ermittelt mit Rechnerstruktur und -features vermittelt und zwischen widerstreitenden Anforderungen und Designmerkmalen abwägt.**
- ***Resultat,*** **das Ergebnis der Entscheidungen, die entstandene "Gestalt" im Vergleich zu anders strukturierten und gestalteten Systemen; der nutzerspezifische Bauplan, der die unterschiedlichen Anforderungen mehr oder weniger kunstvoll integriert hat, aber auch der erarbeitete Satz von "guiding principles", die der Detailkonstruktion zugrunde liegen.**
- ***Qualitätsmerkmale einer Computerstruktur*** **aufgrund eines konsistenten Designs, ausgewogener Dimensionierung und hervorragender Nutzungseigenschaften**
- ***Bestimmte Phase*** **zu Beginn des Entwicklungsprozesses, die grundlegende Planungsphase, zugleich aber auch durchgängige "considerations" und die Evaluation von Design-Systementscheidungen während der gesamten Systementwicklung.**
- ***Momente des systematischen Hardware-Softwareentwurfs*** **im Sinne einer Explizierung von Designzielen, einer Strukturierung des Designprozesses und der Ermöglichung arbeitsteiliger Vorgehensweisen.**
- ***Ansätze einer Problemstruktur,*** **eines Relationennetzes interdependenter Designmerkmale, die simultan oder in enger Abstimmung zwischen den Angehörigen des zentralen Entwicklerteams entschieden werden müssen.**

Trotz der gemeinsamen Entwicklungsanstrengungen war der Architekturbegriff während der Entstehung des Systems und der Beiträge des Bandes (1957 bis 61) noch keinesfalls im Designerteam geschweige denn in der IBM akzeptiert. Die anderen Autoren und Co-Designer wie Erich Bloch, Werner Buchholz und Gerrit A. Blaauw, begnügten sich weiter mit den traditionellen Bezeichnungen wie "machine organization" bzw. der "organization of the computer", dem "logic design" oder dem "computer system design". Auch der Projektleiter, Steve W. Dunwell, hatte 1956 bei der ersten Vorstellung des Vorhabens die besonderen "design objectives" und die "technological and organizational goals" wie "performance, reliability, generality" dargelegt, aber noch keinen neuen Begriff für die grundlegend veränderte Bauweise gefunden. Auch Brooks selber vermied anfangs außerhalb des IBM-Umfeldes in eigenen Büchern und Statusreports über die Computer-Entwicklung sowie in Aufsätzen mit anderen Autoren den Begriff "architecture", er sprach hier stattdessen meist von "com-

puter organization" bzw. "fundamental principles of logical organization".[34] So war das Kapitel "Architecture Philosophy" in dem Stretch-Buch offenbar mehr Ausdruck persönlicher Vorstellungen als offizielle Firmenauffassung, was auch durch die recht distanzierte Erwähnung des Architekturvergleiches im Überblick des Bandherausgebers Buchholz zum Ausdruck kam. Nach der beinahe essayhaften ersten Vorstellung des "concept of architecture" führte Brooks seine Ideen in dem mit Kenneth E. Iverson verfaßten Lehrbuch "Automatic Data Processing" von 1963 näher aus, allerdings ohne das neue "catchword" zu verwenden. In dem Grundsatzvortrag "The Future of Computer Architecture" auf dem IFIP-Kongreß von 1965 trug Brooks dann seinen Architekturbegriff erstmals einer internationalen Fachöffentlichkeit vor, doch dem war die offizielle Adaptation des Begriffs in der neuen IBM 360-Systemfamilie im Vorjahr vorausgegangen.

Brooks zielte mit dem Begriff "system architecture" nicht auf eine Rechnerstruktur- und Bauformenlehre, sondern auf ein neues Rollenverständnis der Computerdesigner und eine gewandelte Designauffassung. Die zentrale Aufgabe des Architekturdesigns war es danach, alle beim Entwurf eines Computersystems auftretenden grundsätzlichen Benutzeranforderungen, Designkonflikte und Allokations-Entscheidungen im Zusammenhang zu betrachten, auf der Grundlage des vorhandenen Erfahrungsschatzes zu bewerten und zu einem konsistenten Gesamtentwurf hoher Qualität zu integrieren. Im Mittelpunkt seines Architekturkonzepts standen dabei ganz eindeutig die Benutzer, d. h. damals die Programmierer, und ihre Anforderungen an das System. So eröffnete er seinen Beitrag im Stretch-Band mit der viel zitierten programmatischen Definition: "Computer architecture, like other architecture, is the art of determining the needs of the user of a structure and then designing to meet those needs as effectively as possible within economic and technological constraints. Architecture must include engineering considerations, so that the design will be economical and feasible; but the emphasis in architecture is upon the needs of the user, whereas in engineering the emphasis is upon the needs of the fabricator." (Brooks 1962, S. 5) Gemessen an den späteren Rechnerarchitektur-Konzepten war Brooks früher Begriff insgesamt noch recht umfassend und heterogen. Er verknüpfte eine ganze Reihe von explizierten und mitschwingenden Bedeutungen, deren Spannweite von der Designphilosophie bzw.-

---

[34] Brooks 1963, S. 45, 47; Brooks hielt sich hier offensichtlich an die Bezeichnungen der vorausgegangenen Berichte über der Stand der 'Rechnerarchitektur', vgl. Lawless 1959, S153, 161 ("computer logical organization", "basic organization of computers"); Beckman; Brooks 1961, S.53 ("logical organization").

methodik über Qualitätsansprüche bis zur professionellen Interessen-artikulation reichte.

Das Brooks'sche Architekturkonzept intendierte zum einen eine Neuordnung der Zuständigkeiten und Designmethoden bei der Rechnerentwicklung. Was bislang mehr als ineinandergreifende Aktivitätsschwerpunkte kooperierender Teams verstanden wurde, sollte nun in zwei zentrale Aufgabenkomplexe aufgeteilt werden: in die übergeordnete und vorgelagerte Funktionsermittlung und -gestaltung einerseits und die nachgelagerte Logik- und Hardware-Implementierung als Ausführung der zuvor definierten Architektur andrerseits. Der Architekturbegriff verweist so auf ein Top-down-Phasenmodell mit der gestalt- und strukturbildenden "architectural phase" am Beginn des Entwicklungsprozesses. Architektur bezeichnete zugleich auch das Ergebnis der Planungsphase, der nutzerspezifische Bauplan und die "architectural philosophy", die in Form eines Satzes von "rationales" und ”guiding principles” als verbindliche Konstruktionsleitlinien die konzeptionelle Integrität auch bei vielen Mitwirkenden sichern sollten.[35] Architektur war insofern ein Instrument der Organisation der Produktentwicklung, sie enthielt Momente des systematischen Hardware-Software-Entwurfs im Sinne einer Explizierung von Designzielen, einer Strukturierung des Designprozesses und der Ermöglichung arbeitsteiliger Vorgehensweisen. Doch weitaus stärker als solche Engineering-Aspekte im Architekturkonzept betonte Brooks den Kunstcharakter des Designs, die kunstvollen Designmethoden und reichen Konstruktionserfahrungen einer professionellen Elite, die über das Wissen verfügt, um die nutzerseitigen Funktionsanforderungen mit Rechnerstrukturen und -features zu vermitteln und dabei zwischen widerstreitenden Anforderungen und Designmerkmalen abzuwägen. Der Begriff implizierte so auch eine Problemstruktur, ein Relationennetz interdependenter Designmerkmale, die simultan oder in enger Abstimmung mit den beteiligten Akteuren entschieden werden müssen.

Architektur kennzeichnet zum anderen aber auch die erreichte *Designqualität*, die Erfüllung aller nutzerseitigen Anforderungen, insbesondere die Klarheit des Entwurfs und den gestaltenden Stil des Chefarchitekten bzw. des Designerteams, die "design consistency and conceptual maturity that characterize the ingenious architect" (Brooks 1965, S. 88). Brooks kennt zwar auch den neutralen deskriptiven Architekturbegriff, der etwa die "architecture of Stretch" von

---

[35] "The universal adoption of several guiding principles helped ensure the conceptual integrity of a plan whose many detailed decisions were made by many contributors." (Brooks 1962, S. 7)

anders gestalteten Systemen und Bauweisen abhebt, doch meistens hatte Architektur für ihn eine normativ-wertende, qualitätsvergleichende Bedeutung. Dabei sind für ihn nicht einzelne Designmerkmale wie Rechengeschwindigkeit oder Funktionsfülle qualitätsentscheidend, sondern die konzeptionelle Integrität des Ganzen, d.h. die am Hardware-Software-Interface erkennbare Konsistenz der Nutzungseigenschaften des Rechners: "I will contend that conceptual integrity is *the* most important consideration in system design. It is better to have a system omit certain anomalous features and improvements, but to reflect one set of design ideas, than to have one that contains many good but independent and uncoordinated ideas." (Brooks 1975, S. 42)

Die Geschlossenheit des Designs war für Brooks aber nicht durch Team- oder Kommissionsentscheidungen herstellbar, sondern allein durch die Gestaltungskraft der "senior architects". Er forderte deshalb auch den Übergang von der üblichen horizontalen Arbeitsteilung auf Teambasis zur hierarchisch organisierten vertikalen. Die Aufteilung der Systementwicklung in die Architekturdefinition durch die Chefdesigner und die deren "set of philosophies and design principles" folgende Implementierung durch die "engineers" wurde für ihn sogar der entscheidende Ansatz zur Erreichung der konzeptionellen Integrität und Güte eines Systems (Brooks 1975, S.44). In der Folgezeit bildete sich bei Brooks eine ausgeprägt aristokratische Anschauung aus, wonach der Fortschritt im Computer- und Systemdesign allein von einer kleinen professionellen Elite getragen wird. Wie die Werke der großen Künstler und Komponisten seien die wenigen Spitzenprodukte das Werk von herausragenden Einzelpersönlichkeiten oder von sehr kleinen Designteams mit "really first class minds": "great designs come from great designers." (Brooks 1987, S. 15).

Die "system architecture" erstreckte sich deshalb bei Brooks nicht auf ein bestimmtes technisches Arbeitsgebiet, sondern auf alle die komplexen Aufgaben, für die besondere Integrations- und Syntheseleistungen notwendig waren. Die eigentliche Zuständigkeit des Architekten war so das "management of complexity" (Spruth 1976, S. 14), und dessen Aufgabenfelder veränderten sich im Laufe der Reifung der einzelnen Technikdisziplinen. So konstatierte Brooks eine inhaltliche Schwerpunktverschiebung in der architektonischen Systemgestaltung von der bis ca. 1960 dominierenden Hardware-Strukturierung, der "pure computer architecture" zur "software architecture" und "architecture of input/ output systems", die er beide auch als "phenomenology" des Computersystems zusammenfaßte. Das künftige zentrale Aufgabengebiet der "architects" sah er stärker im Design des Gesamtsystems, in der "integrated hardware and software

architecture" bzw. in der noch anwendungsnäheren "information system architecture" (Brooks 1965, S. 88). "System Design" und ”Computer” bzw. ”System Architecture” waren so vom Prinzip her eine "advancing discipline", die einen so hohen Komplexitätsgrad aufwies, daß dafür keine adäquaten Theoriemodelle und Methoden einer "complete analysis" zur Verfügung stünden. Denn Architektur war für Brooks wesentlich durch nicht einfach zergliederbare Komplexität, durch unvollständiges Wissen und eine Kombination heterogener Elemente geprägt. Sie beruht daher im Unterschied zum "engineering" nicht auf sicheren Regeln, sondern auf Erfahrungswissen, das nicht zerlegt oder logisch abgeleitet werden könne: "Design, therefore, requires art as well as science, judgement as well as knowledge. Each new application poses or emphasizes difficulties previously unknown or ignored.” (Brooks, Iverson 1963, S. 437).

Während Brooks im reinen Hardwarebereich durch Methoden und Erfahrungsfortschritte ein exakteres, stärker berechenbares Wissen erwartete, kam er 1975 in seiner kritischen Bestandsaufnahme des Software Engineering und noch deutlicher in Aufsätzen aus den Jahren 1977 und 1986 zu dem Resultat, daß die Komplexi-tät bei der Konstruktion von anwendungsnahen System- und Informations-Architekturen sowie von ”software objects” nicht vorübergehender, sondern prinzipieller Natur sei: "Our complexities are arbitrary, because they are fruits of many independent minds acting independently. [...] This complexity is compounded by the necessity to conform to an external environment that is arbitrary, unadaptable, and ever changing." (Brooks 1987, S. 6) Software Engineering, System Design und System Architecture folgen seiner Meinung nach daher nicht dem Paradigma der Mathematik oder der Physik. Eine Überwindung von Komplexität durch abstrakte Modellierung scheitere ebenso wie der Versuch, Heterogenität und Vielfalt auf eine ”fundamental unified theory” zurückzuführen. Brooks hielt deshalb am Ende den Begriff der ”Computer Science” selbst für eine irreführende, die Zunft fehlleitende Bezeichnung (Brooks 1977, S. 625).

Außer auf designmethodische bzw. –theoretische Aspekte zielte Brooks Architekturbegriff aber von Beginn an auch auf den professionellen Status der Systemdesigner und ihre Stellung in der betrieblichen Arbeitsteilung. Brooks sprach ausdrücklich von der Etablierung der Computerarchitektur als einer "professional speciality" um ca. 1960 und dem Aufstieg einer besonders qualifizierten Gruppe von 2-3 Dutzend "senior architects", deren Qualifikationen sich grundlegend von den früheren Tüftlern und Entwicklern von ”machine concepts” mit immer höheren Leistungen und ökonomischeren Bauweisen

unterschieden (Brooks 1965, S. 87 f.). Die Fähigkeiten der "new profession" der Computerarchitekten lagen für ihn vorrangig im Erkennen der Bedürfnisse der Benutzer und die Vermittlung dieser Anforderungen mit den technischen und ökonomischen Restriktionen zu einem konsistenten und reifen Design: "So it is not to much to say that the past five years have seen the birth of a profession. The computer architect designs the external specifications, gross data flow and gross sequencing of a system. He is, like the building architect, the user's advocate. He must balance the conflicting demands of engineer (cost, speed), programmer (function, ease of use) and marketeer (function, speed, cost) to yield the machine of greatest true value to the user" (ebda, S. 88). Das Wissen dieser "senior architects" finde sich in keinem Lehrbuch, sondern sei in vielleicht 50 wertvollen Architekturtraktaten herausragender Computerdesigner aufgezeichnet. Das Können beruhe vor allem auf eigenen langjährigen Designerfahrungen, auf dem intensiven vergleichenden Studium anderer Systeme und einer Klassifizierung der Designprobleme wie der "principles, philosophies, and priorities", die zu ihrer Lösung bereitstünden. Diese herausragende Profession von Systemarchitekten der größeren Computerhersteller war nach Brooks nach dem Auslaufen der hochschuleigenen Rechnerentwicklungen der eigentliche Träger der Innovationen im Computerbereich geworden.

Die Einführung des Begriffs der ”Computer Architecture”, die Hypostasierung ihres Kunstcharakters und die Anlehnung an den professionellen Status von Bauarchitekten sollten die gewachsene Komplexität der Aufgabenstellung eines stärker nutzerorientierten Hardware-Software- und Systemdesigns kennzeichnen und auch die gewachsenen Qualitätsansprüche und das berufliche Selbstbewußtsein einer Designerelite ausdrücken. Daneben artikulierte der emphatische Architekturbegriff auch professionelle und soziale Aspirationen einer kleinen Schicht von Chefdesignern. Als Anwalt der Benutzerbedürfnisse und -interessen und als Designkonflikt-Manager wollte Brooks die ”senior computer architects” etwas aus der betrieblichen Arbeitsteilung herausnehmen und ihnen in den Hersteller-Kunden-Beziehungen einen Sonderstatus zubilligen, ein Anspruch, der vielleicht direkt mit der deutlich verstärkten Einflußnahme der Unternehmensleitung auf die divergierenden Designphilosophien und Produktlinien-Strategien der führenden IBM-Entwicklungszentren in Verbindung stand (Pugh u. a. 1991, S. 191)

Der Architekturbegriff, der zunächst nur Brooks persönliche Design- und Systemphilosophie artikulierte, wurde bald auch von den engsten Kollegen rezipiert und stieg in dem Zeitraum 1961 bis 64 zu einem IBM-weit akzeptierten

Markenzeichen auf. Dabei erfuhr er aber eine deutliche Bedeutungsverschiebung, so daß neben die individuellen Architekturkonzepte einzelner Chef-Designer eine offizielle Version trat. Nach der ersten Verwendung in Lyle R. Johnsons IBM Research Report "A Description of Stretch" tauchte im Jahre 1960 in ebenfalls internen Denkschriften von Brooks und Blaauw über die von beiden gemeinsam entwickelte IBM-8000er Großrechnerfamilie ein erweiterter Begriff auf. Zusätzlich zum funktionalen Nutzen kennzeichnete ″architecture″ nun auch die besonderen Qualitätseigenschaften des Produkts und besonders das einheitliche Erscheinungsbild einer Systemfamilie gegenüber dem Nutzer (Johnson 2000; Pugh u.a. 1991, S. 137). Brooks und Blaauw beeinflußten sich in der folgenden Zeit wechselseitig bei der Erweiterung und Systematisierung des Architekturkonzeptes, doch setzten beide durchaus eigene Akzente. Gerrit A. Blaauw, wie Brooks vor seinem IBM-Engagement Doktorand bei Aiken, betonte in Vorlesungen ab 1967, in Aufsätzen ab 1970 und in seinem Buch ″Digital System Implementation″ von 1976 stärker die hierarchische Strukturierung des Designprozesses, die Verwendung formaler Methoden und die Systematisierung von Designprinzipien. In seinem Dreistufen- bzw. Dreiphasenmodell sollte die ″architecture″ das funktionale Verhalten des Systems festlegen, die ″implementation″ dies in einen logischen Strukturentwurf übersetzen, die die ″realization″ dann in die physikalische Struktur transferiert (Blaauw 1970; 1972). Die Architekturvorgabe durch den oder die Chefdesigner diente einem ″controlled design process″, bei dem die Eigenkoordination der unterschiedlichen Expertenteams zurückgedrängt wird, ohne dabei bestimmte Designentscheidungen vorzuschreiben.

In seinem bekannten, auf Anregung von Heinz Zemanek 1972 für die "Elektronischen Rechenanlagen" verfaßten Architekturtraktat ″Computer Architecture″ bemühte Blaauw sich erstmals auch um eine Systematik der Designprinzipien von Systemarchitekturen, die letztlich den Kern der ″quality of architecture″ ausmachen. Die Konsistenz einer Architektur beruht danach auf den Hauptprinzipien Orthogonalität, Angemessenheit und Verallgemeinerung sowie daneben auf Symmetrie, Sparsamkeit, Transparenz, Kompatibilität, Offenheit und Vollständigkeit. Einige der Kriterien übernahm Blaauw dabei direkt aus dem Vitruvschen Forderungskatalog für gute Architekturentwürfe.[36] Die jeweilige Bedeutung der Prinzipien für ein ″good design″ demonstrierte er an historischen und aktuellen Beispielen, ohne dabei aber das Problem zu

[36] Persönliche Mitteilung von Heinz Zemanek. Danach geht der Rekurs auf Vitruv auf Blauuws Lehrer van Wijngaarden zurück.

erörtern, wie aus den z. T. gegenläufigen Kriterien ein konsistenter Entwurf hervorgeht (1972, S. 156-159). Die Korrektheit und Effizienz des dreistufigen "translation process" wollte Blaauw vor allem durch die Anwendung formaler Sprachen bei der Spezifikation von Architektur, Systemkonfiguration und Hardware gewährleisten. Mit seiner stärkeren Betonung der hierarchischen Strukturierung, Modularisierung und Formalisierung als Strategien des "mastering complexity" stand er offensichtlich unter dem Einfluß der Strukturierungskonzepte seines Landsmannes Dijkstra.

Blaauw und Brooks begannen bereits seit der zweiten Hälfte der 70er Jahre, ihre Konzepte zu einer geschlossenen Architekturlehre zu integrieren, erste Kapitel zirkulierten bereits 1978 in der Community (Bell, Mudge, McNamara 1978, S. 25), doch abgeschlossen wurde dieses Vorhaben erst in ihrem gemeinsamen großen Werk "Computer Architecture" von 1997. Darin verknüpfen sie das nutzerbezogene Qualitätskonzept, die Designkriterien-Systematik, das Dreistufenmodell als Basis der vertikalen Arbeitsteilung mit dem Ebenenkonzept und einer hierarchisch-strukturierten Beschreibung der Architekturvarianten nach Vorbild des "design-space concept" von Gordon Bell und Allen Newell von 1971. Das Problem der Lösungskombination und der Synthese zu einem konsistenten Design behandeln sie dann nicht mehr systematisch, sondern exemplarisch anhand herausragender historischer Computer: "Here one can see the examples in detail and in context. Each machine is an integrated collection of decisions, and the integration itself demands study." (Blaauw, Brooks 1997, S. 499). Blaauw und Brooks lassen damit nach einer Periode der Verwissenschaftlichung des Architekturkonzeptes und der Bauformensystematik sowie der quantitativen Leistungsbewertung die erfahrungsbasierte komparatistische Bewertung von Meisterwerken der Computerentwicklung wieder aufleben. Sie bringen so das von ihnen einst entwickelte ursprüngliche Konzept der Computerarchitektur als Resultat der Austarierung von Qualitätskriterien, von professionellen Designkonflikten und sozialen Entscheidungssituationen sowie von individuellen Stildifferenzen der gestaltenden Personen wieder in die aktuelle Architekturdebatte ein.

Die zunehmende Resonanz, die der Architekturbegriff in der zweiten Hälfte der sechziger Jahre erfuhr, beruhte jedoch nicht allein auf den persönlichen Architekturkonzepten von Brooks und Blaauw, so anerkannt sie als Angehörige der IBM-Designerelite und, ab 1965, als Professoren der Computer Science in Chapelhill bzw. Twente auch sein mochten. Es war vielmehr erst die Verknüpfung des Begriffs mit der wohl erfolgreichsten Produktentwicklung der

Computergeschichte, dem IBM-System/360, die den endgültigen Durchbruch in der Computer Community bewirkte. Zwischen 1961 und 1964 wurde ”Architecture" zu einem auch von der IBM-Spitze anerkannten Leitbegriff der neuen einheitlichen Systemfamilie, der sich durch deren Welterfolg in der darauffolgenden Zeit auch schnell außerhalb des Unternehmens ausbreitete. Architektur wurde dabei schnell gleichbedeutend mit dem "common concept" und "style" einer Geräteserie. (Zemanek 1980, S. 4) Doch unverkennbar verlagerte sich der Bedeutungsschwerpunkt dabei auch von der Designphilosophie und -methodik zum Entwicklungsmanagement und vor allem zu Kompatibilitätsaspekten hin.

Bereits 1961 tauchte der Architekturbegriff in dem Endbericht der von Brooks geleiteten SPREAD-Kommission auf, die den Gesamtplan für die Zusammenführung der bisher getrennt laufenden wissenschaftlichen und kommerziellen Rechnerlinien und für eine Vereinheitlichung der IBM-Produktpalette entwickelt hatte. Unter ”architecture” wurde dabei ein Satz von ca. 40 Grundregeln verstanden, mit dem die Grundlinien der ”logical structure” festgelegt wurden, insbesondere die ”downward bzw. upward compatibility”, die Adressierung und die Operationsprinzipien sowie eine Reihe von Qualitätsmerkmalen. Die Architektur erhielt dadurch unmittelbar Leit- und Lenkungsfunktion in der arbeitsteiligen Produktentwicklung, auf ihr beruhte die funktionelle Spezifikation der Architektur und die folgende Detaillierungsphase, wobei die konsequente Einhaltung durch eine ”permanent systems architecture group” sowie ein ”architectural-engineering design office” kontrolliert werden sollte.[37] Zentrale Designkriterien waren demgemäß die Generalisierung der Funktionen ("general purpose function"), die "intermodel compatibility" und die Antizipation absehbarer Technologietrends ("open-ended-design"). Zugleich sollte die Architektur der ”360-Familie” durch die strikte Trennung des äußeren Erscheinungsbildes gegenüber dem Nutzer (”logical structure”) von der jeweiligen technischen Umsetzung (”physical structure”) die durchgängige Kompatibilität über die gesamte Systemfamilie hinweg garantieren.[38] Umgesetzt wurde dieses Ziel durch die relative Adressierung, die erstmals systematische Nutzung der Mikroprogrammierung, die konsequente Systemmodularisierung mit Standard Interface und Kanalkonzept für periphere Informationsflüsse sowie mit Hilfe

[37] Haanstra u.a. 1961, S. 6 f.11 ff.; Evans 1983. Die Schilderung der Entstehung des System /360 von dem Projektmanager Bob O. Evans (1986, S. 168 ff.) zeigt deutlich, wie die einzelnen Entwicklungsphasen und -stäbe tatsächlich über die architektonischen Leitlinien gesteuert wurden.

[38] Das Wiederaufgreifen der von Wilkes geschaffenen Methode der Mikroprogrammierung bot die technische Grundlage für die Herstellung einer so durchgängigen Kompatibilität.

eines einheitlichen Betriebssystems (Ganzhorn,. 1997, S. 48-50). Die Fokussierung auf eine von der jeweiligen Hardware-Implementierung der einzelnen Typen unabhängige Nutzungskonformität der Systemfamilie gewann mit der öffentlichen Ankündigung des Systems/360 im April 1964 die Oberhand, ”architecture” wurde mehr und mehr zum Inbegriff der unternehmensstrategischen Produktpaletten- und Systemfamilienpolitik. Der ”system architect” war damit nicht mehr vorrangig Anwalt *individueller* Nutzerinteressen, sondern er wurde zum Garanten der Hardwarekompatibilität und Softwareportabilität über ein großes Spektrum unterschiedlicher Systemkonfigurationen hinweg.

Das Konzept der Architektur als Blaupause für eine ganze Rechnerfamilie wurde nach den außerordentlichen Erfolgen der IBM von vielen Computer-herstellern imitiert (Lipovski, Doty 1978, S. 54). Doch die nachhaltigste Wirkung dieses Architekturansatzes ging nicht von den Kompatibilitäts-Aspekten aus, folgenreicher war vielmehr die Fokussierung der Definition von Amdahl, Blaauw und Brooks (1964, S. 87) auf den Instruktionensatz und die Nutzungsqualitäten in der Perspektive der Programmierer: "The term architecture is used here to describe the attributes of a system as seen by the programmer, i. e., the conceptual structure and functional behavior, as distinct from the organization of the data flow and controls, the logical design, and the physical implementation." Diese Definition und Sichtweise wurde in der Folgezeit breit rezipiert [39], die IBM-Designer-Elite wirkte so mit ihrem gestalterischen Interface-Konzept der Computerarchitektur ausgesprochen schulebildend in der sich etablierenden neuen Technikdisziplin. Diese Richtung hatte das große Verdienst, die Aufmerksam-keit der bislang Prozessor-fixierten Entwickler besonders auf das "interface" des Systems zum "user" und die konkreten Nutzungseigenschaften sowie auf funktionsgemäße Gütekriterien zu richten. Doch zugleich leistete sie einer Einengung auf eine bloße "Fassadensicht" der Architektur Vorschub: durch die Abtrennung des Befehlssatzentwurfes als hierarchisch höher verortetes Architekturdesign von der nachgeordneten Implementierung und Hardware-Realisation als weniger anspruchsvollen Engineering-Aufgaben geriet der systemische Zusammenhang zwischen Befehlssätzen, Rechnerorganisation und Hardwareressourcen aus dem Blick.

---

[39] So 1966 von M. Flynn 1969 von F. G. Heath und in den 70er Jahren dann u. a. von Abrams/Stein, Spruth, Lipovsky/Doty, Dennis/ Fuller/ Ackermann, J. Hayes, Doran und Garside.

Kennzeichen des Architekturbegriffes der 60er Jahre

- gewachsene Komplexität des Gesamtsystems von Hardware und Software
- vermehrte Lösungsalternativen und Designkonflikte.
- integrative Synthese-Leistungen in der Entwicklung
- keine Ingenieurprinzipien und theoriebasierten Dekompositionsstrategien verfügbar
- Gestaltung der Gesamtfunktion und des Mensch-Computer-Interface
- Austarierung heterogener Anforderungen zu einem konsistenten Gesamtentwurf
- aristokratische Designkultur
- hierarchische Organisation des Entwicklungsprozesses
- Abhängigkeit von der Designmanagement-Qualifikation der "Chief-Designer"

Die Trennung von äußerem Erscheinungsbild der Rechnerfunktionalität und interner technischer Umsetzung stieß daher in den 70er Jahren zunehmend auf Kritik, vor allem in der Parallelrechner-Community (z. B. Bode, Händler 1980, S. 1-3) und in der pragmatisch-quantitativen Architektur-Bewertung, die in den 80er und 90er Jahren im Rahmen der RISC-Technologie hervortrat. Der Konzentration auf die Schnittstelle zwischen Hard- und Software setzten die erfolgreichen RISC-Entwickler John L. Hennessey und David A. Patterson deshalb später die Forderung entgegen, Hard- und Software als ein Kontinuum und nicht als getrennte Teile zu betrachten: "Die Optimierung eines Maschinenentwurfs muß über alle diese Ebenen erfolgen. Dazu muß man mit einer Vielzahl von Technologien, angefangen bei Compilern und Betriebssystemen, bis hin zum Logikentwurf und zur Gerätetechnologie vertraut sein" (Hennessey, Jouppi 1980; Hennessey, Patterson 1994, S. 13, Hennessey 1999, S. 28) Im Zuge der Etablierung der Computerarchitektur seit den 70er Jahren wurde bei einem Großteil der "architectuaral community" der Begriff daher nach und nach wieder auf die Rechnerorganisation und den Hardware-Bereich ausgeweitet: "Computer architecture has now become a popular term and some computer architects have expanded its meaning to include studies in machine organization, network structure, system evaluation, and system software. This less strict interpretation of architecture concerns itself with implementation techniques and computer organization." (Lipovski, Doty 1978, S. 54).

Heinz Zemanek, der in dem Wiener IBM-Forschungslaboratorium den Architektur-Diskurs der IBM-Spitzendesigner weiterführte, machte hier die

wichtige Unterscheidung zwischen der äußeren und inneren Architektur, zwischen den "Architekturen der Schnittstellen" und der "Gesamtarchitektur", die "in engem Zusammenhang stehen und ein Ganzes bilden". Davon wiederum unterschied er die "Abstrakte Architektur" als gebiets- und disziplinübergreifende Qualitätsnormen-Lehre und Gestaltungsmethodik. (Zemanek 1978; 1980 S. 23 ff.; 1986, S. 108) Auch bei Zemanek ist das Kernthema die Komplexitäts-beherr-schung im Computerdesign bei gleichzeitiger Erfüllung nutzerbezogener Qualitäts-ansprüche. Zemanek bündelt seine Überlegungen zu einer Theorie des Systementwurfs und darüber hinaus zu Ansätzen einer allgemeinen Gestaltungslehre technischer und organisatorischer Systeme. Die Grundlage bietet ihm dabei ein Vier-Phasenmodell der Architekturkonzepte: die intuitive und minimalistische Phase der Pioniere, die durch Arbeitsteilung und Spezialisierung geprägte Baumeister-Phase sowie die Komplexitätssteigerung der Architektonischen Phase, deren Leitmotiv der Turmbau zu Babel wird. Zemanek sieht ähnlich wie die anderen Architekturtheoretiker der IBM die Lösung in einer formalisierten Top-Down-Architekturkonzeption wie in einem systematischen Merkmalskatalog „guten Designs“. Doch geht er über dieses, von ihm „abstrakte Archi-tektur“ genannte Konzept hinaus, indem er die Notwendigkeit einer weiteren Phase postuliert, der Rehumanisierung des Entwurfs, die in einem Rückbezug auf die konkreten Nutzer und die betroffenen Akteure im Alltag des Gebrauchs und in einer Rückbesinnung auf die klassische Klarheit der Designprinzipien der Pioniere besteht. Architektur ist so am Ende für Zemanek immer aus Fleisch und Blut und nie als abstraktes Struktur- und Normengebilde vorab zu definieren. Aus seiner Architekturlehre leitete er schließlich ein interdisziplinäres Ausbildungsprogramm und eine Berufsethik der Informatik ab: Um den Anforderungen eines veranwortungsvollen Systemgestalters gewachsen zu sein, sollte der "Computer- bzw. Systemarchitekt" neben formalen Methoden, mathematischen und physikalischen Kenntnissen auch über geistes- und sozialwissenschaftlichen Problemwissen verfügen. Das normative Architekturkonzept von Blauuw und Brooks ging so am Ende in einem integralen Ansatz der soziotechnischen System- und Organisationsgestaltung auf.

Rechnerarchitektur als deskriptive Erfahrungswissenschaft und Engineering-Methodik

Deskriptive Bauformensystematik der Computersysteme: Bell, Newell 1970/71, Blaauw, Brooks, 1997

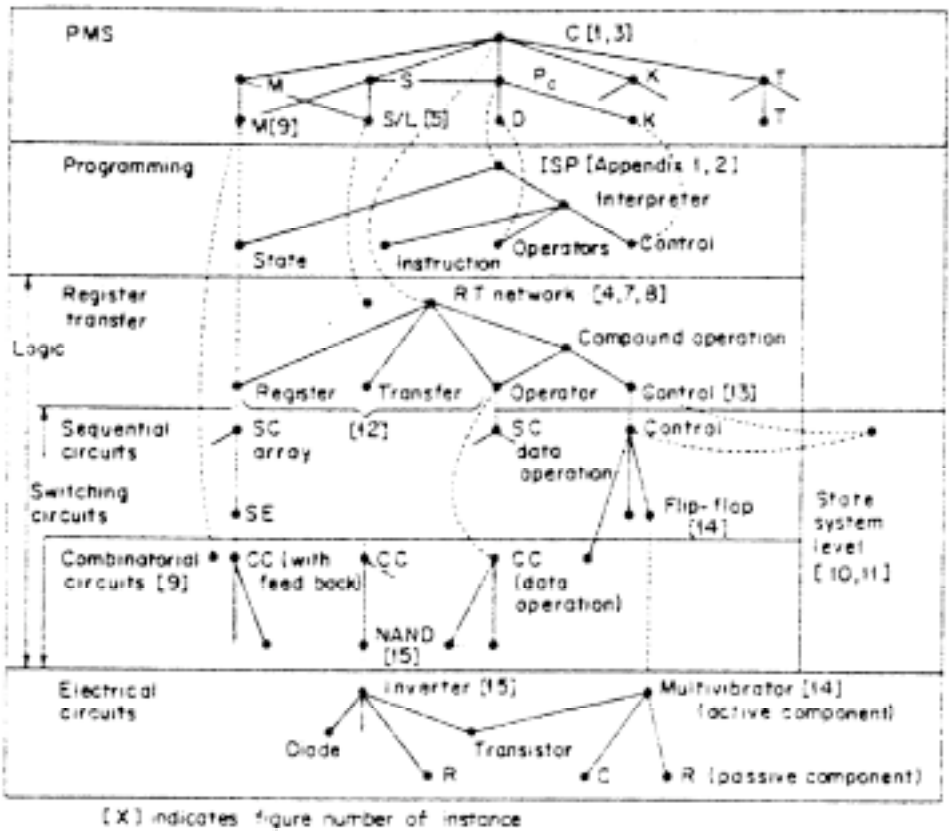
- Sammlung von Architekturtraktaten herausragender Computerdesigns
- Entwicklung einer Taxonomie von Hardware/Softwaresystemen auf Basis einer Hierarchie von "well defined system levels" (Vorbilder: Linnés Klassifikation und Parnas SODAS-System)
- Hierarchische Modellierung des gesamten "design space" bzw. "computer hyperspace" zur Sichtbarmachung historisch realisierter Lösungsalternativen und von "possible design choices"

Hierarchische Schichtenkonzepte der Rechnerarchitektur: Dennis, 1972, Tanenbaum, 1976/84/90, Hayes, 1978, Myers, 1978/82, Baer, 1980, de Blasi, 1990

- Verallgemeinerung der Schichtensysteme für Betriebssysteme auf das gesamte Computersystem; Konzept der "structured computer organization" nach Vorbildern der "multilevel virtual machine" von Dijkstra u.a.
- Übertragung des strukturierten Software-Entwurfs auf die Designphasen des Rechnerentwurfs
- Vorrang der Komplexitätsreduktion vor der Lösungsraum- Erschließung: Übergang von der historisch-deskriptiven Systematik zur logischen Ordnung funktionaler Schichten

Modularisierungkonzepte der Rechnerarchitektur: Baskin, Roberts, 1972, Fuller, Sieworik, 1973, Stone, 1975

- Übertragung des Bauelemente-Montage-Modells auf die Gesamtarchitektur: systemtechnische Baukastenmodelle
- Rationalisierung des Konstruktionsprozesses nach dem Vorbild der Fertigung und des Austauschbaus



Hierarchische Designraum-Darstellung der PDP-8-Architektur von Bell, Newell 1971, S. 127

5. Die Etablierung der Computerarchitektur als Erfahrungswissenschaft: Designraum-Modellierung in hierarchischen Strukturen

Der Architekturbegriff bündelte während der 60er Jahre eine ganze Reihe von Trends im Rechnerentwurf und in der Designer-Community. Er kennzeichnete die gewachsene Komplexität des Gesamtsystems von Hardware und Software und die daraus resultie-renden vermehrten Lösungsalternativen und Designkonflikte. Er apostrophierte gegenüber den Baukasten- und Elemente-Montage-Modellen des vorausgegangenen Engineering-Konzeptes die integrativen Synthese-Leistungen in der Entwicklung, für die noch keine Ingenieurprinzipien und theoriebasierten Dekompositionsstrategien zur Verfügung standen. Dazu gehörten vor allem die Gestaltung der Gesamtfunktion und des Mensch-Computer-Interface sowie die Austarierung heterogener Anforderungen zu einem konsistenten Gesamtentwurf. Die Lösung des Syntheseproblems wie die intendierte hohe nutzeradäquate Designqualität wurde aber vor allem von einer aristokratischen Designkultur und einer hierarchischen Organisation des

Entwicklungsprozesses erwartet, damit aber auch, ähnlich wie beim ebenfalls bei IBM entstandenen Chief-Programmer-Ansatz in der Softwarekonstruktion, von den Integrationsleistungen und der Designmanagement-Qualifikation der "Chief-Designer" abhängig gemacht. Mit dieser Personalisierung und Ästhetisierung des Designproblems ließ sich zwar ein Professionalisierungsanspruch begründen, nicht aber die Konstituierung einer technischen Disziplin. Es entwickelte sich daher um 1970 eine Richtung in der "architectural community", die zwar am "architectural model" als Grundlage des Designmanagements festhielt, doch die sich zugleich gegen eine zu starke Betonung des Kunstcharakters des Designs und gegen das überhöhte Leitbild des Allround-Systemarchitekten verwahrte. Gegenüber der Hoch- bzw. Überschätzung qualitativer Aspekte setzte man hier mehr auf quantitative Methoden, auf eine systematische Erfassung und Strukturierung des Designraums, die Entwicklung entsprechender Notationen und Werkzeuge sowie besonders auf die Deskription von Designkonflikten. Damit erhielt der Computerentwurf zwar noch keine theoriebasierte wissenschaftliche Grundlage, doch es entstand bis zum Ende der 70er Jahre eine deskriptive Erfahrungswissenschaft, die in Teilbereichen bereits auf eine Rationalisierung des Designs zielte.

Der emphatische Architekturbegriff wurde jedoch beibehalten, er entwickelte sich schnell zum Leitbegriff der neuen Disziplin. Es häuften sich die Belege für die Begriffe Systemarchitektur und Computerarchitektur in Tagungsbänden und Zeitschriften. Als Schlüsselbegriff erschien "architecture" erstmals 1969 in Lehrbüchern über Computerdesign, 1970 folgten gleich mehrere Darstellungen und das erste Buch mit dem Titel "Computer Architecture" (Foster). Im gleichen Jahr erschien in den Proceedings der nationalen Computerkonferenzen der USA erstmals die Rubrik "Systems Architecture" anstelle der früher üblichen "Computer Organization". 1972/73 erfolgte dann der erste Durchbruch bei der Institutionalisierung: 1972 bildete sich bei der ACM eine "Special Interest Group Architecture" (SIGARCH), die ab 1973 die "Computer Architecture News" bzw. ab 1978 den "SIGARCH Newsletter" herausgab. Parallel dazu entstand bei der Computer Society der IEEE das "Technical Committee on Computer Architecture". Beide zusammen organisierten ab 1973 das "Annual Symposium on Computer Architecture", das bald den Zusatz "International" erhielt und das bis heute die wichtigste Fachkonferenz des Gebietes geblieben ist. Im Jahrzehnt nach der Verkündung der neuen Profession des Systemarchitekten durch Brooks auf der IFIP-Tagung von 1965 war so der Übergang von einer exotischen Metapher für dezidierte Designphilosophien und dem Markenzeichen für eine Rechnerfamilie zu einer allgemein akzeptierten

Bezeichnung für ein wissenschaftliches Forschungsfeld und akademisches Lehrgebiet vollzogen.

Der erfolgreiche Einstieg in die Professionalisierung und die Institutionalisierung des neuen Gebietes in den Departments für Computer Engineering bzw. Computer Science konnte freilich nicht verhindern, daß mit "architecture" in der Community recht unterschiedliche Intentionen und Wissenschaftskonzepte verknüpft wurden. Die Architekturtraktate und Lehrbücher in der ersten Hälfte der 70er Jahre zeigen überhaupt noch eine große Unsicherheit gegenüber dem neuen Leitbegriff und weisen sehr große Unterschiede hinsichtlich der Reichweite des Gebietes, der theoretischen Zielrichtung und der Methodenbasis auf. Daran änderten auch die Bemühungen der ACM SIGARCH und des IEEE Technical Committee on Computer Architecture um eine verbindliche Definition nur wenig, zumal sie beträchtlich voneinander abwichen. Der Vorschlag des IEEE-Komitees war nämlich sehr Hardware-orientiert und ging eher in die Richtung des klassischen Engineering: ”Computer architecture is the study and design of algorithms and logical control for the management of the physical resources of a computer system” (zit. nach Abrams, Stein 1973, S. 67). In der ACM-Gruppe legte der Professor für Computer Science an der Universität Maryland Yaohan Chu demgegenüber eine viel breitere, mehr an Frederick Brooks angelehnte Definition vor: ”Computer architecture is an emerging discipline as a result of recent rapid technical advances in computer technology. Instead of merely studying the hardware structural and algorithmic features of a computer system, the scope of computer architecture is being broadened to the conceptual formation and specification of computer systems. [...] Before the end of this decade, a new breed of professionals, computer architects, will arise. Computer architects practice computer architecture just as building architects practice building architecture." (zit. nach Bell 1973, S. 18). Die vorgeschlagenen Definitionen für das Fachgebiet spiegelten mithin nur den Gegensatz zwischen einer mehr Gestaltungs- und einer mehr Engineering-orientierten Architektur-Auffassung wider, der die gesamte junge Disziplin durchzog.

So gab es Architekturlehren, die die Brooks'sche Architekturmetapher ganz in das Zentrum stellten, so Michael Flynn, Boris Beizer, Marshall D. Abrams, Philip G. Stein und Earl C. Joseph. Michael Flynn (1966, S. 97 ff.), der Autor des meistverwendeten Klassifizierungsschemas für Rechnerarchitekturen (1972), knüpfte direkt bei Brooks an und entwickelte ein entsprechendes

Szenario der künftigen Forschungsaufgaben des neuen Gebietes. ”Architecture” stand dabei vor allem für ein Nutzer-zentriertes ”system design”, für eine komplexe, alle Umgebungsfaktoren einbeziehende Sicht des Entwicklungsprozesses: ”The entire realm of physical resources, integrated with the systems resources we call computer architecture, should be directed toward furthering the best possible solution of the computer user's problems." (ebda., S. 97, 103). Die zentralen Forschungsaufgaben des Gebietes sah Flynn demgemäß in dem Mensch-Maschine-Interface, der internen System-System-Kommunikation und schließlich in der Schnittstelle zwischen dem System und den Anwendungen. Flynn dehnte damit den von Brooks und Blaauw auf die Programmierer eingeengten User-Begriff auf alle Systemnutzer aus.

Der als F&E-Direktor bei Data System Analysts mit vielen Rechnersystemen vertraute Boris Beizer übertrug in seinem zweibändigen Werk "The Architecture and Engineering of Digital Computer Complexes" von 1971 den Brookschen Architektur-Ansatz auf das Design von Multiprozesser-, Multicomputer- und Rechnerverbundsystemen. Die herausragende Aufgabe des Architekten im Unterschied zum Systems Engineer bestand für ihn in der Vermittlung zwischen verschiedenen Akteurs- und Experten-Perspektiven sowie in der Integration der disparaten Hardware-, Software-, Engineering- und Ökonomie-Entscheidungen zu einem von ihm ermittelten "optimum design": "We can summarize the architect's role by stating that it is synthetical, catalytic, and translative. His design is a synthesis of the substance of subordinate disciplines. He motivates and directs the efforts of those disciplines by giving their practitioners a point of view tempered by considerations outside of their domain. He is a linguist, a polyglot in a tower of Babel." (Beizer 1971, Bd. 1, S. 10 f., Bd. 2, S. 836) Die Ausführung seiner Architekturlehre besteht dann auch neben der Designraum-Beschreibung besonders in der Darlegung von Design-Alternativen und Tricks, mit denen sich eine optimale Balance zwischen den widerstreitenden Zielen und Designkonflikten bei Multicomputer-Konfigurationen herstellen läßt. Beizer demonstriert dies besonders anhand der Austarie-rung der Zielkonflikte beim Bau hochverfügbarer Computer, beim „viability design“, das er selber Ende der 50er Jahre in der militärischen Systementwicklung praktiziert hatte. (ebda, Bd. 2, Kapitel 11 u. 12, bes. S. 625) Außer auf die Multiperspektivität hat der Architekt auch vorrangig auf die Langzeitperspektive von Computer Complexes zu achten, er denkt in Zeiträumen eines Jahrzehnts, während die „system engineers“ sich lediglich um momentane Belange von Rechnersystemen zu kümmern haben. Sie sind verantwortlich für die genaue

„execution of the architects's plan", doch wollte er dies ausdrücklich nicht als ein Unterordnungsverhältnis verstehen: „ The engineer is not subordinate to the architect." (ebda., S. 836)

Abrams und Stein erwarteten ganz ähnlich in ihrem Addison-Wesley-Lehrbuch von 1973 von der Mikroelektronik eine massive Zunahme der Wahlfreiheiten und damit der Designkomplexität und "trade-offs" zwischen Hard- und Software-Design, die nur noch von einem vermittelnden Systemintegrator zu bewältigen wären: "Just as an architect [...] so must the computer designer balance requirements, user interface, and costs to make a viable design. [...] The architect, to some extent takes the final responsi-bility for the way in which this money is spent, and his systems concept can make the difference between a mediocre machine and an immensely successful one. Many of the necessary decisions can be made logically, but the overall problem is complex enough to require an artistic solution." (ebda., S. 67) Auch Earl C. Joseph verknüpfte in seinem Gesamtüberblick über die "Systems Architecture" in dem Wiley-Lehrbuch für "Computer Science" von 1972 den Brooksschen funktionalen Architekturansatz mit der LSI-Entwicklung. Für den Manager der Militärrechner-Abteilung von Univac verschob sich der Schwerpunkt von den Spitzendesigns der Architekturelite zur alltäglichen Systemkonfiguration quasi-individueller Rechner aus massenproduzierten LSI-Standardmodulen. Das Architekturdesign hatte sich deshalb unmittelbar den Problemen der Nutzer und Trends der Gesellschaft anzupassen, ja die "creative art" und zugleich "science-based technology of computer system architecture" wurde zum wichtigen gesellschaftlichen Aktionsfeld: "Computer system architecture can no longer be disengaged from society's values. [...] As a result, the computer is finding wider applic-ation in the social areas for solving man's real problems." (Joseph, 1972, S. 104, 150)

Gegenüber diesen Gestaltungs-betonten Architektur-Auffassungen sahen andere Autoren in "computer architecture" nicht viel anderes als die gewohnte Engineering-zentrierte "organization" der Hardware oder primär eine Designmethode für Rechnerfamilien. So lehnte sich das Architekturkonzept von Frederick G. Heath stark an den IBM-Schnittstellen- und Systemfamilienansatz an. Heath, der seit dem Ende der vierziger Jahre bei Ferranti militärische Digitalsysteme entwickelte und als Chefingenieur von Ferranti-I.C.L. in Manchester für das Design der 1900er-Serie der I.C.L. verantwortlich war (Heath 1977, S. 141 ff.), grenzte in seinem Lehrbuch "Digital Computer Design" von

1969 das allein die Hardware betreffende ″computer design″ von der ″computer architecture″ ab, die die ″overall logical structure″, die ″operational procedures", die ″programming arrangements″ und alle Nutzungsaspekte betraf, wobei dies alles aber mit dem Hardware-Design rückgekoppelt werden mußte. Unter dem Einfluß von Brooks und Dijkstra entwickelte Heath ein achtstufiges Phasenmodell, die ″hierarchy of computer design and architecture stages″, die mit dem Architekturdesign (Nutzergruppen, Spezialsoftware, Compiler) begann und über das Betriebssystem, den Instruktionssatz zur Hardware-Implementierung ging (Heath 1969, S. 109 ff., 124). Der enge Bezug zur Rechnerfamilien-Entwicklung ließ hier die Strukturierungsaspekte gegenüber den Nutzer-bezogenen Designqualitäten in den Vordergrund rücken, was auf eine deutliche Prägung der Architekturauffassung durch die Akteurskonstellation schließen läßt.

Andere Autoren benutzten den neuen Leitbegriff, weil er ihren Professionalisierungs-Aspirationen mehr entsprach, ließen sich aber nicht auf neuartige konzeptionelle Überlegungen ein. So waren etwa für Herbert S. Sobel, der sich in seinem 1970 bei Addison Wesley erschienen Lehrbuch ″Introduction to Digital Computer Design″ auch besonders auf Phister und Flores berief, ″computer architecture″ und ″computer organization″ identisch. Sie bezeichnen lediglich elementare Prozessorkonfigurationen, während sehr viele Architektur-Entscheidungen wie ″special or general purpose–computer", serielle oder parallele Betriebsweise, externe Plugboard-Programmierung oder interne Speicherprogrammierung als Probleme der von der Architektur gesonderten funktionalen, operationalen und programmtechnischen Charakteristiken angesehen werden. In der Ausführung verschwimmen dann aber die vorher so sorgsam entwickelten Klassifizierungsmerkmale. Besonders bei der Behandlung der Probleme der ″control synthesis″ gerät Sobels letztes Ziel, die Entwicklung systematischer Methoden der Handhabung der ″computer requirements, architecture and design″ vollends ins Wanken: Wegen der wechselseitigen Abhängigkeit von ″organizational structure″ und ″instruction set″ sei eine klare Folge der Entwicklungsschritte nicht möglich, beide müßten ständig aufeinander bezogen und wechselseitig modifiziert werden: ″Thus it becomes apparent that the architecture of a computer is a convergent process which may, in fact, reflect the ideosyncrasies or ingenuity of a designer or design group." (Sobel 1970, S. 222).

Während Sobel immerhin noch mit der konzeptionellen Einordnung des Architekturbegriffes rang, wurde dieser von anderen Autoren lediglich als aktuelles Schlagwort aufgegriffen und eher äußerlich auf eine relativ konventionelle Engineering-Sicht des Computerdesigns als Bausatzmontage aufgesetzt. Zu dieser Sichtweise fühlten sich diese nicht zuletzt durch den von der IC-Entwicklung ausgelösten Standardisierungsprozeß bei den Rechnerkomponenten bestärkt. So setzte etwa Thomas C. Bartee in der 1972 erschienenen 3. Auflage des McGraw-Hill-Lehrbuches "Digital Computer Fundamentals" "computer architecture" einfach mit dem in den früheren Auflagen von 1960 und 1966 verwendeten "system organization of digital computers" gleich: "This area is often referred to as computer architecture, for it is the broad and general design which is considered rather than the details of circuit construction, interconnection technics, etc." (S. 365, S. XI). Bei Caxton C. Foster, der 1970 als erster "Computer Architecture" als Titel wählte, tauchte der Begriff sogar nur in der Überschrift und im Vorwort auf, während im Text nach wie vor die etablierten Begriffe "computer organization" bzw. "configuration" verwendet wurden. Foster, seit 1966 Professor für Computer Science an der University of Amherst, Mass., nutzte das Flair des neuen Leitbegriffes offensichtlich zur nachträglichen Erhöhung des Stoffes der Vorlesung über "comparative machine design", das die eigentliche Grundlage des Buches bildete. Architektur war für Foster wie für viele Autoren in der Etablierungsphase nur eine Residualgröße, nämlich all das, was über die beschriebenen Grundlagen und Bausteine des Rechnerentwurfes hinausging und insofern noch keine Wissenschaft, sondern die Kunst des integrierenden Designs.[40] Das erste Buch mit dem Titel "Computer Architecture" war alles andere als eine elaborierte Design- oder Architekturlehre.

In den Folgejahren erschienen noch eine Reihe ähnlicher Lehrbücher, die den Begriff "Computerarchitektur" nahezu ausschließlich im Titel oder im Vorwort verwendeten. So z. B. die Einführung von Eike Jessen aus den Jahre 1975 (S. 14 ff., 65 ff.), die erste deutsche Darstellung mit dem neuen Leitbegriff, die vom Typ her eine Zusammenstellung aller für den Rechnerentwurf nützlichen system- und netzplantechnischen Verfahren, quantitativen Methoden sowie Strukturierungskonzepten und Interpretationsschemata darstellt, darunter eine originelle Graphen-Notation zur Modellierung des Rechners als eines Auftrags-,

---

[40] In diesem Kontext findet sich ein Rekurs auf Brooks: "The field of computer architecture, or the art of designing a machine that will be a pleasure to work with is only gradually receiving the recognition it deserves." (S. IX).

Rechen- und Funktionseinheiten-Systems. Demge-genüber repräsentiert M. Morris Manos bei Prentice Hall ebenfalls 1975 erschiene-nes Lehrbuch den Typ des Modulbaukastens von den einfachsten zu den komplexesten Komponenten. "Computer architecture" war für Morris bereits ein etablierter Begriff, "commonly used to denote the organization and design of digital computers" (ebda., S. XI), der den Autor zu keinerlei grundlegenden Reflektionen mehr reizte.

Waren derartige Lehrbücher mit ihrem explorativen, kompilatorischen oder traditionellen Charakter noch Ausdruck des Frühstadiums der neuen Disziplin, so wiesen andere Architekturlehren und -traktate bereits den Weg zur Verwissenschaftlichung. Diese Gruppierung von Computer Scientists bzw. Engineers gab sich weder mit einer oberflächlichen Rezeption des Begriffes zufrieden noch wollte sie sich in dem Richtungstreit zwischen dem Engineering- und Kunstverständnis der Rechnerarchitektur einer der beiden Seiten zuordnen lassen, ihr ging es vielmehr um eine Überwindung dieses Gegensatzes durch eine theoretisch-methodische Ausfüllung des Architekturkonzeptes. Diese Autoren arbeiteten zum Teil aufgrund von Anleihen bei der Software-Strukturierung hierarchische Schichtenmodelle und Modularisierungskonzepte zur Erfassung und Beschreibung des Designraums aus, sie schufen erste Bauformensystematiken und Analysemethoden für Trade-off-Situationen beim Design und für Bottlenecks in Rechnern und legten so die Grundlagen für die Weiterentwicklung der Computerarchitektur zu einer deskriptiven, vergleichenden Erfahrungswissenschaft, die den Wissensschatz der Designer zusammenstellte, die "possible design choices" systematisch ordnete, ohne dabei bereits den Anspruch einer "comprehensive theory of computer systems" zu erheben (Bell, Newell 1971a, S. 38).

Das herausragende Werk in der Etablierungsphase der Disziplin war ohne Zweifel das Werk "Computer Structures. Readings and Examples", das der Systemarchitekt der Firma DEC, C.Gordon Bell, während einer Gastprofessur an der Carnegie Mellon-Universität mit dem KI-Pionier und Modellbildungs-Experten Allen Newell zusammen zwischen 1968-71 verfaßte. Angelegt als eine Kombination einer Strukturlehre und Klassifikation von Computersystemen mit einer Traktatsammlung zu ausgewählten Musterarchitekturen, war das Buch eine Synthese aus langjähriger Designpraxis und dem technikwissenschaftlichen Bestreben um eine Systematisierung von Rechnerstrukturen und -bauformen. Den praktischen Entstehungshintergrund bildeten Bells Erfahrungen beim

Design fast aller Computer und Minirechner der Firma Digital Equipment von der PDP-1 von 1959 bis zum Grunddesign der PDP-11 im Jahre 1970. Die schnelle Folge der Entwicklungsprozesse, die Notwendigkeit arbeitsteiliger Designabläufe und das Problem der Hardware- und Software-Kompatiblität der PDP-'Systemfamilie' hatten Bell veranlaßt, von der uncodifizierten Architektur im Kopf des Chefdesigners zur genauen Spezifikation und Strukturierung der Systeme in einem "architectural model" überzugehen (Pearson 1992, S. 93 f.)

Die Blickerweiterung auf das sich durch die IC-Entwicklung schnell ausbreitende Marktsegment der Minirechner führte bei Bell zu einer weitaus bescheideneren Auffassung der Rolle des Systemdesigners. Ohne Brooks namentlich zu nennen, kritisierte er dessen Stilisierung des Designers zum Anwalt der Benutzer und zum Allround-Architekten: "It is not clear that every user can afford an architect, especially when computer costs are very small, nor is it clear that one can stand the unreliability of one-of-a-kind computers. The activity of building an information processing system will more likely be collecting computer components, the associated basic operation systems and programming languages, and than interconnecting them and programming the resulting structures of the components [...] The evolution of the architect as an all-knowing mediator to create a globally optimum system for each application may not occur." (Bell 1973, S. 18 f.) Auch die Bewertung der Rolle einzelner Architekten für die Gesamtentwicklung der Rechner fiel bei Bell und Newell weitaus nüchterner aus als bei Brooks: nicht große Architekten bestimmen für sie den Gang der Computer-Geschichte, sondern eher wie in der biologischen Evolution eine Kombination vieler kleinerer unpersönlicher Entwicklungsschritte[41]. Diesem nicht-heroischen Geschichtsverständnis entsprach auch die zurückhaltende Verwendung des Architekturbegriffs: Bell und Newell nutzten ihn zwar gelegentlich in den seit 1970 erschienen Aufsätzen für den "style" eines Rechners oder für die Hardware-Software-Schnittstelle einer Rechnerkomponente, doch selten als zentralen Begriff. In ihrem Buch sprachen sie sogar fast ausschließlich von "computer structure", erst in den Architekturtraktaten der zweiten Hälfte der 70er Jahre und in der stark erweiterten Neubearbeitung von 1982 wurde auch bei ihnen "architecture" als Gebietsbezeichnung und Systemcharakteristik voll akzeptiert.

---

[41] „[...]the history of computer systems is not just a story of particular men discovering or building particular things, but of a somewhat more impersonal and whidespread series of advances that have changed computer systems radically." (Bell, Newell, 1971, S. 39)

Die Vorbehalte gegenüber der Architekturmetapher bedeuteten indes nicht, daß Bell und Newell beim Rechnerdesign den Status einer exakten, theoriebasierten Technikwissenschaft in absehbarer Zeit für erreichbar hielten. Denn im Unterschied zum Schaltkreisentwurf, wo der Zustand der "systematic art" erreicht und so eine Design-Rationalisierung möglich sei, würde der Reifungsprozeß bei den Rechnerstrukturen auch noch nach 30 Jahren der Entwicklung immer wieder durch technische Fortschritte unterbrochen: "However, the immaturity is dictated [...] by the shifts in technology that continually throw us into previously uninhabited parts of the space of all computer systems. Whatever systematic techniques start to emerge are left behind." (Bell, Newell 1971b, S. 387) Der Architektur-Designer hat es deshalb nicht mit einem eingrenzbarem Lösungsraum, invarianten Grundprinzipien und festen Relationen zwischen meßbaren Designparametern zu tun, sondern mit einem sich permanent erweiternden "design space" und "newly emerging trade-off regions", für die er immer wieder neue Organisationsstrukturen finden muß (ebda., S. 394). Das konkrete Design von Computern war dazu in hohem Maße durch betriebliche und gesellschaftliche Umgebungsfaktoren bestimmt, die die "constraints" des Entwurfs und das "system of individual design trade-offs" ständig verschieben. "A computer", so resümierte er in einer nachträglichen designmethodischen Auswertung der von ihm geleiteteten Entwicklungen bei DEC, "is a product of its total environment" (Bell, Strecker 1976, S. 1; Sieworek, Bell 1982, S. XIV).

Bell und Newell stimmten damit zwar mit Brooks überein, daß sich die Designkomplexität von Computern durch äußere Einflußfaktoren und daraus folgende interne Ungleichgewichte immer wieder neu aufbaut. Doch im Unterschied zu diesem vertrauten sie nicht auf den genialen Integrator und hierarchische Organisationsformen beim Entwicklungsmanagement. Sie verlagerten den Ansatzpunkt der Komplexitätsbewältigung vielmehr auf die hierarchische Strukturierung des Konstruktionsgegenstandes, die empirische Erfassung des "design space"- und eine gezielte Methodenentwicklung für alle Ebenen des Computersystems. Da ihnen einerseits das Methodenangebot der Kunst- bzw. Architekturlehre wie Anforderungs- und Gütekriterienkataloge nicht ausreichten und sie andrerseits nicht glaubten, daß sich die internen Beziehungen zwischen den Rechnerkomponenten nach dem Vorbild der mathematisch-naturwissenschaftlich fundierten Technikwissenschaften formalisieren und theoretisch begründen ließen, entwickelten Bell und Newell ihre Lehre nach dem Muster der deskriptiv-klassifizierenden Disziplin der Botanik. So war ihr Vorbild für die Designraum-Modellierung das Linnésche Klassifikationssystem, ihr Buch sollte deshalb auch ursprünglich "Computer Botany" oder "Computer Taxonomy" heißen.

Grundlage der Designraum-Erfassung ist bei ihnen eine rein deskriptive Notation, die den gesamten historisch realisierten und erschlossenen ″computer space″ in eine Hierarchie von fünf ″well defined system levels″ in aufsteigender Reihe von der physikalischen (circuit level) über die logische (logic und register-transfer level) und symbolische Ebene (programming level) zur obersten Komponentenstruktur (processor-memory-switch level) darstellt (Bell, Newell 1971a, S. V, 3-12). Das Ebenenmodell wird dabei in Anlehnung an Dijkstrasche Strukturierungsmethoden und das SODAS-System von Parnas (1967) - über bloße Klassifikationsebenen hinausgehend - als eine Kette von Abstraktionsebenen verstanden: "[...] each level arises from abstraction of levels below it. Each does a job that the lower levels could not perform because of the unnecessary detail they would be forced to carry arround" (ebda., S. 3). Wie bereits Ende der 40er Jahre bei Stibitz angedacht, wurde nun in Anlehnung an Software-Strukturierungsmethoden die hierarchische Schichtung des Gesamtdesigns zur entscheidenden Methode, die Designkomplexität zu minimieren. Auf der Basis des Ebenenmodells, der ″organized hierarchy of many levels″, werden dann die verschiedenen "design choices" als ein vielfach gestaffeltes Baum- und Verzweigungsmodell der Designvarianten entwickelt. Dieses mündet, ergänzt durch zusätzliche Nutzungs- und Gestaltungskriterien, in einem universalen Raummodell des Designkosmos. Als Darstellungsmodell dient hier ein Mehrvektorenmodell herausragender "computer space dimensions", das nach Nutzungstypen, Mengen- und Komplexitätsgraden geordnet ist. In diesem vorhandenen bzw. möglichen Designraum werden die konkreten Computersysteme als Kombination zwischen den speziellen Funktions-, Leistungs- und Strukturmerkmalen bestimmt: ″It is useful to think of all the dimensions as making up a large space in which any computer system can be located as a point.″ (Bell, Newell 1971a, S. 38).

Diese höchst differenzierten Modellbildungen konnten zwar endlich die älteren simplen Evolutionsmodelle der Rechnerstammbäume ablösen. Doch ließ die Vermengung historisch-praktischer und technisch-systematischer Klassifikationskriterien dabei aber keinen "orthogonalen Aktionsraum" des Designers [42] entstehen, der die Grundlage für eine wissenschaftliche Designmethodik hätte bilden können. Auch ein strukturierendes Designphasenmodell war bei der multiperspektivischen Vielfalt der Raummodelle kaum möglich. Da mit zunehmendem Detaillierungsgrad der Klassifikationsmerkmale die Zahl der Lösungs-

[42] Diesen Begriff verwendet Max Syrbe (1995, S. 222) in einem neuerlichen Designraum-Ansatz.

varianten explodiert und die komplette Designraumbeschreibung angesichts der mannigfaltigen Interdependenzen keine Hilfe mehr bei der Lösungsselektion bietet, waren Bell und Newell letztlich doch gezwungen, die eigenen erfahrungsbasierten Bewertungen des historisch realisierten "computer space" zugrundezulegen. Die Taxonomie sollte dem Computerarchitekten jedoch bei gezielten Variationsbildungen und Überschreitungen des traditionellen Designraums behilflich sein. Der Überschaubarkeit wegen konzentrierten sie die bis 1970 verwirklichten ca. 1000 Computertypen auf 40 signifikante Musterarchitekturen, die sie in Form von Architekturtraktaten und exakten Beschreibungen mit der von ihnen geschaffenen Notation im zweiten Teil des Werkes ausbreiteten. Der "case study approach" in den "Compu-ter Structures" wie auch in der folgenden Reevaluation der DEC-Produktlinien war für Bell die logische Folge der mangelnden Theoretisierbarkeit des Rechnerdesigns: "Because no theory exists to undergird this multidimensional design problem, we believe that there is no substitute for an extensive, critical understanding of the existing examples of designed and marketed systems." (Bell, Mudge 1978, S. VIII)

Der historisch vergleichende Ansatz erhielt bei ihnen so einen zentralen Stellenwert, da Designer nirgendwo besser als anhand früherer Lösungsbeispiele die Integration widersprüchlicher Anforderungen und Designkriterien studieren könnten.[43] Bell und seine Mitarbeiter sahen in ihren Arbeiten eine unmittelbare Fortführung des Buchholzschen Buches "Planning a Computer System" von 1962 und ein Pendant zu der im Entstehen begriffenen Architekturlehre von Blauuw und Brooks, die in dem zweiten Teil ("Computer Zoo") tatsächlich auch ein sehr ähnliches Darstellungskonzept aufweist. Dieses Werk erschien jedoch, wie oben behandelt, erst 1997 als Nachzügler der 'Historischen Schule' der Computerarchitektur, die ihren Höhe- und Endpunkt in den beiden Auflagen der "Computer Structures" gefunden hatte. In der Kombination von Klassifikationsansatz, Designraumbeschreibung und Fallstudien konkreter Musterexemplare bildete die Bell-Newellsche "comparative taxonomy of design choices" (Sieworek, Bell 1982, S. XIV) nämlich den disziplingeschichtlich entscheidenden Übergang von der Meisterwerke-vergleichenden Kunstlehre zu den designwissenschaftlichen Bestrebungen für exakte Taxonomien und Algebren der Rechnerarchitektur in den 80er und 90er Jahren.

---

[43] "For those of us who must deal with design goals, constraints, and objective functions to improve reliability, availability and maintainability, it is imperative that we first clearly understand previous design problems." (Bell, Mudge 1978, S. IX)

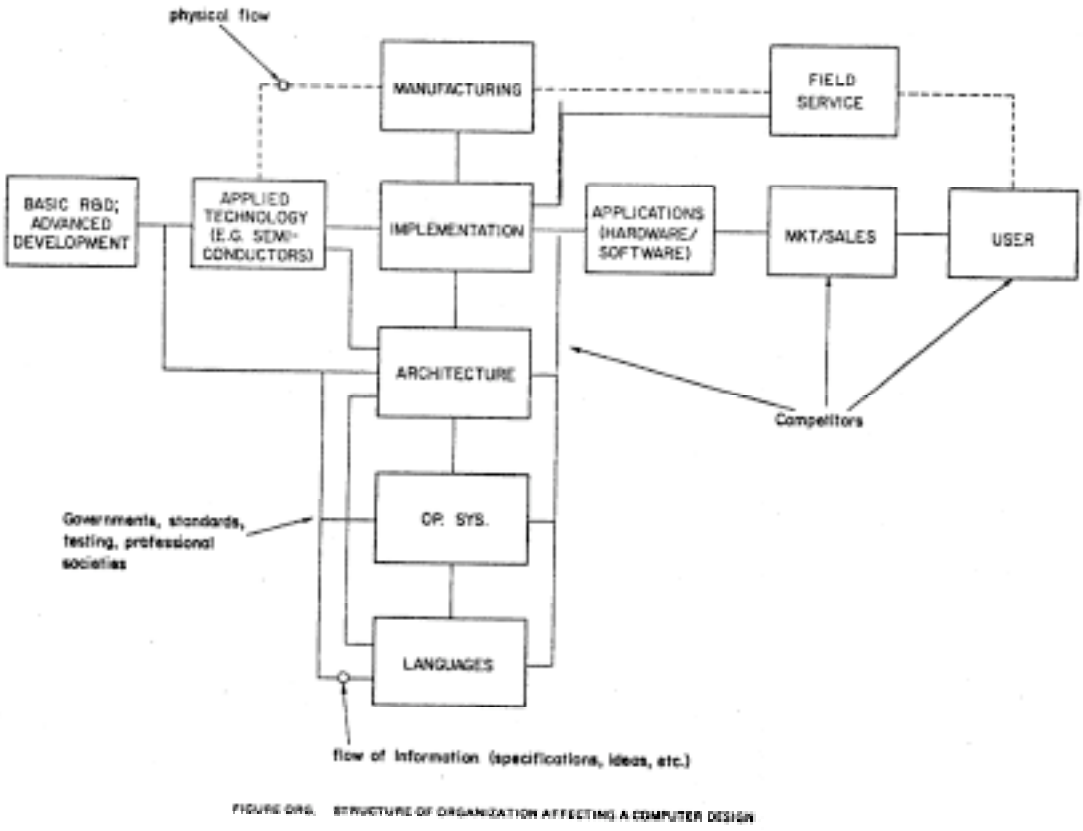


FIGURE DRG. STRUCTURE OF ORGANIZATION AFFECTING A COMPUTER DESIGN

Organisationsstruktur und Informationsflüsse beim Computer-Design nach Bell

Bei der Methodenentwicklung bemühten sich Bell und Newell, die technik- oder strukturbedingten Änderungen in den Korrelationen der Designparameter mit quantitativen Modellen zu erfassen. Doch die verwendeten Leistungsbewertungs-Verfahren für verschiedene Dimensionierungs-Entscheidungen waren nach eigener Einschätzung noch völlig unzureichend: "There are no precise measures for performance [...] it is difficult to predict what effect changing even the most direct structural variables will have (e. g., memory speed)" (Bell, Newell 1971, S. 52) Erst in der Neubearbeitung von 1982 wurde mit dem "Kiviat graph", der sechs zentrale Parameter in einem radialen Schaubild zusammenfaßt, ein Bewertungstool vorgestellt, mit dem die "balance of a system" anschaulich dargestellt werden konnte, so daß man auf Daumenregeln für Einzelgrößen verzichten konnte (Sieworek, Bell 1982, S 46). Doch Bells Studien sahen für komplexere Designfragen in quantitativen Ansätzen nur ein begrenztes Potential, da die Anzahl der Variablen beim Entwurf eines Computersystems zu hoch ist, zu viele Wechselbeziehungen zwischen ihnen bestehen und dazu noch unberechenbare dynamische Faktoren und "real world constraints and opportunities" einen Einfluß ausüben. Bell fächerte die

Einzelkomponenten der Kosten- und Leistungsberechnung über den ganzen Produktlebenszyklus hinweg auf und gelangte dabei zu einem komplizierten Netzwerk der Designentscheidungen. Dessen Komplexität nimmt, wie es Stibitz schon einmal erkannt hatte, durch die Einbeziehung der Nutzungsphase und der Qualitätsaspekte, der "user life cycle costs" wie der "product life cycle costs" sogar noch deutlich zu, da mit dem "user designing, repairing, remodeling, redesigning, retiring" immer neue Designanforderungen und -kriterien hinzukommen. Mit Kurvenvergleichen und mathematischen Modellen ist der Designkomplexität eines kompletten Produktlebenszyklus nicht Herr zu werden, denn in dem "multivariable mathematical problem" des Computerdesigns ergibt sich das Gesamtoptimum nicht aus der Summe gewichteter Einzeloptima, sondern nur durch die Suche nach vernünftigen Kompromissen: "It is too simplistic to think that computer design should be a well-defined activity in which mathematical programming can be employed to obtain optimum solutions." (Bell, Mudge 1978, S. VII, 22)

Aus der Heterogenität der Anforderungen und der Vielzahl der Problemdimensionen und möglichen "design styles" zogen Bell und seine Mitstreiter die Konsequenz, daß eine einzige Modellsicht auf das Rechnersystem nicht ausreicht. Dieses läßt sich nicht allein als Struktur- oder Funktionsebenen-Hierarchie beschreiben, als Interpretations-Kette, Baukasten-Komponentenstruktur oder Schnittstellenkonzept. Es sind vielmehr mehrere Modellsichten möglich und nötig, wobei die Gegenüberstellung die jeweiligen Modellsichten, deren Potentiale und Grenzen sichtbar macht und so davor bewahrt, die Modellperspektive mit der höchsten Affinität zur eigenen Organisationsstruktur bzw. Profession zu verabsolutieren. (ebda., S. 2-25). Die gleiche Multiperspektivität wird selbst bei der Betrachtung der verschiedenen Ebenen des Computersystems erwartet: "Each level is characterized by a distinct language for representing the components associated with that level, their modes of combination, and their laws of behavior." (ebda., S. 2) So ist beim "circuit level" die klassische technikwissenschaftliche Sicht der Elektrotechnik notwendig, denn hier gelten wirklich *Gesetze* wie die von Ohm oder Kirchhoff, während beim "logic level" nur der Digitaltechnik eigentümliche Modellrepräsentationen und Sichtweisen erforderlich sind, die die Systeme durch diskrete Variablen und logische Funktionen modellieren. Der "program level", der eigentlich eine "entire technology in its own right" darstelle, bedarf wiederum einer nur für Computer charakteristischen linguistischen Betrachtungsweise und entsprechenden Repräsentationsformen. Die oberste Ebene schließlich, erfordert eine der chemischen Verfahrenstechnik entsprechende Modellsicht, da es hier um

kontinuierliche Prozesse und Fließgrößen geht, für die Flußdiagramme, Warteschlangen- und Gleichgewichtsmodelle die entsprechenden Darstellungsmittel sind (Bell, Newell 1971a, S. 3-10). Für die Computertechnik ist so ein ganzer "set of conceptual levels" bestimmend, der sich nur schwer zu einem einzigen technikwissenschaftlichen Theoriesystem integrieren und nach einer einzigen Leitdiziplin modellieren läßt, nicht einmal nach einer deskriptiven wie der Biologie.

Gerade deswegen erhielt die Betrachtung von "trade-offs" eine Schlüsselrolle in der bereits "life cycle"-orientierten Architekturlehre Bells und seiner Mitautoren. Sie erkannten die große Lücke bei den qualitativen Methoden, insbesondere zur topologischen Beschreibung von Designkonflikten, die sich aus den unterschiedlichen Vorgaben der Akteure ergaben: ”The design tools we have for discussing (and discovering) appropriate designs are weak [...] in particular, there is no really useful language for expressing the trade-offs in a rough and qualitative way, yet precise enough, so that the design consequences can be analyzed.” (ebda., S. 394) Bell und Newell waren mit dieser Einsicht im Bereich des Rechnerentwurfes zu der grundlegenden Methodenlücke vorgedrungen, auf die in der Mechanik-Konstruktion bereits Hugo Wögerbauer in den 40er Jahren in seiner "Technik des Konstruierens" gestoßen war (Hellige 1995, S. 146-49). Sie ließen es aber weitgehend bei der verbalen Konfliktbeschreibung bewenden und griffen hier weder Konzepte der Polyoptimierung (u.a. Peschel, Riedel 1976) und des "Constraint-Managements" noch Ansätze von Christopher Alexander (1964) auf, Designkonflikte mithilfe von topologischen Modellen der "problem's structure" zu lösen. Sie überließen es Autoren wie Abrams/Stein (1973) und David J. Kuck (1978), die akteurs- und problembezogenen Trade-offs im einzelnen an Designalternativen durchzu-spielen. Bells und Newells Hauptinteresse galt letztlich doch weniger der *Designkonflikt-Modellierung* und dem Management von "Stakeholder"-Anforderungen als der *Designraum-Strukturierung* und noch mehr der Erschließung neuer Regionen des Computerdesigns mithilfe der Klassifikation. Denn ihr Architekturkonzept war noch in hohem Maße getragen von dem Bewußtsein eines expandierenden ”computer space” und der Begeisterung für die evolutionäre Vielfalt der Bauformen, selbst für die ausgefallensten Exemplare, die, da möglicherweise Ausgang für neue Entwicklungsrichtungen, nicht dem Vergessen anheimfallen sollten.

Rechnerarchitektur als nahezu exakte und formalisierte Technik- und Designwissenschaft

Exakte Beschreibungssprachen für die Designraum-Klassifikation der: Barbacci 1973/82

- Präzisierung der historisch-deskriptiven Klassifikation von Bell/Newell durch ein exaktes Sprachkonzept mit dem Ziel einer automatischen Exploration des "design space"

Theoriebasierte Taxonomien der Rechnerarchitektur nach naturwissenschaftlichen Vorbildern: Dasgupta 1984/89/90

- Verwissenschaftlichung der Computerarchitektur nach dem Muster der Chemie, um das Design von Zufallsentscheidungen und individuellen Konstruktionserfahrungen unabhängig zu machen
- Abkehr von der historisch-deskriptiven Klassifikation zugunsten einer exakten Taxonomie auf der Basis von "basic concepts" und formalisierten Notationen
- Ablösung des "informal" durch den "formal design process": strikte Ordnung des Konstruktionsprozesses durch vorab definierte Methodenschritte gemäß der Taxonomie des Lösungsraumes und der hierarchischen "levels of abstraction"

Systematisierung der Klassifikation des Lösungsraumes und Methodenintegration mit dem Leitbild der Algebra: Everling 1996

- Mathematisch formalisierte Modellierung des Lösungsraumes der Rechnerarchitekturen: Übergang von der biologischen zur chemischen und mathematischen Klassifikation
- Ergänzung der klassifikatorischen Bauformenlehre durch eine regelbasierte Konstruktionsmethodik und verbindliche Konstruktionsregeln
- Verwissenschaftlichung als Methode der Erfahrungskompression und Designrationalisierung

6. Ausblick und Fazit: Von der Komposition zur Dekomposition und zurück?

Aufgrund ihrer empirischen Reichhaltigkeit, des theoretischen Reflexionsniveaus und des breiten Methodenspektrums wurden die Arbeiten von Bell und Newell ein wichtiger Anknüpfungspunkt für die Architekturkonzepte der 70er bis 90er Jahre. In ihnen verschob sich mit fortschreitender Etablierung und Akademisierung der Disziplin sowie mit der wachsenden Komplexität der Computersysteme die Zielrichtung von der *Ausweitung* zur *Begrenzung* des Lösungsraumes. Ebenso wurde die bisherige Vielfalt der Modellsichten zugunsten weitgehend homogener Betrachtungsweisen und Repräsentationsformen reduziert, um so der Theoriebildung näherzukommen. Die Designkonflikt-Modellierung wurde nun aufgegeben und durch Lösungsstrategien ersetzt, die die Komplexität und Heterogenität vorab ausschalteten. Dabei geriet die Computerarchitektur zunehmend unter den konzeptionellen Einfluß der

Software-Architektur bzw. des Software-Engineering, wo die Komplexitäts-mindernden Strukturierungsmethoden schon seit dem Beginn der 60er Jahre vorherrschend waren. So kam es nun auch zu Ansätzen für eine "structured computer architecture", in denen das hierarchische Schichten- bzw. Modularisierungskonzept ganz in den Mittelpunkt der Architekturlehre rückte. Beim strukturierten Rechnerentwurf ging man immer mehr von der historisch deskriptiven Designspace-Strukturierung zu einer logischen Ordnung funktionaler Schichten über, mit der man weniger auf die Lösungsraum-Erweiterung als auf die Rationalisierung des Computerdesigns zielte (Myers 1978; Hayes, 1978; Dennis u.a. 1979; Doran 1979; Baer 1980).

Einige Autoren betrachteten hierbei den hierarchisch strukturierten Designablauf mehr aus der Softwareperspektive im Sinne einer logischen Kette von Interpretationsstufen ("interpretation hierarchy" bzw. "hierarchy of interpreters", Spruth 1976, S. 16; Bell, Mudge 1978, S. 3 ff.) oder von "virtuellen Maschinen" mit je eigener Maschinensprache (Tanenbaum 1976). Für konsequente Vertreter der "structured computer architecture" erschienen die einzelnen "well-strucured objects" sogar wie Sätze einer strikten Grammatikregeln folgenden Sprache (Boulaye 1976). Andere Autoren hatten aus der Hardwaresicht eine im Vergleich zu früheren Engineeringkonzepten wesentlich elaboriertere Baukasten-Hierarchie vor Augen, bei der das Gesamtsystem aus jeweils in sich geschlossenen Hardwaremodulen und Software-komponenten top-down durch ein "stepwise refinement" entwickelt wird. Das Design wird hierbei in klassischer Ingenieurmanier nach dem Muster des industriellen Austauschbaus und Montageprozesses modelliert, wobei das Konstruktionsobjekt und die Prozeßkette die gleichen hierarchischen Modellstrukturen aufweisen, was die Heterogenität von Designentscheidungen und Designkonflikte gar nicht erst in Erscheinung treten läßt (Fuller, Sieworik 1973; Stone 1975; vgl. Bell, Mudge 1978, S. 3-9).

Andere Richtungen nahmen ihren Ausgangspunkt eher bei der Bell-Newell'schen Bauformen-Systematik. So entwickelten eine Reihe von Autoren seit dem Ende der 70er Jahre mit Blick auf die neuen Architekturen der Parallelrechner oder Datenflußmaschinen die schichtenbasierte Designraum-Beschreibung zu *evolutionären* Taxonomien weiter, die der erwarteten Ablösung der Von-Neuman-Architekturen logische Notwendigkeit verliehen (u.a. Bode, Händler 1980; Giloi 1978, 1981, 1984; Hwang, Briggs 1984). Aus Bells offenem Entwicklungsmodell einer Ausbreitung des Designraumes wurde hier ein gerichtetes Fortschrittsstufenkonzept. Eine weitere Gruppe von Architektur-

lehren bilden die in den 80er und 90er Jahren aufkommenden theoriebasierten Taxonomien, die die Computerarchitektur durch enge Anlehnung an Naturwissenschaften wie die Chemie oder an die Mathematik endgültig zu einer strengen Technikwissenschaft weiterentwickeln möchten (u.a. Dasgupta, 1984; Everling, 1996). Die von Flynn (1972) begonnenen, von Händler (1978), Skillikorn (1988) u.a weitergeführten Klassifikationsschemata werden dabei so feingegliedert und theoretisch untermauert, daß sie ihren Charakter als historisch entwickelten Lösungsraum abstreifen und zu einer Art chemischer Notation aufsteigen, die alle möglichen Elementeverbindungen von der Atom- bis zur Makromolekülebene über Formeln zugänglich macht. Die Verwissenschaftlichung soll hier vor allem helfen, das Design von Zufallsentscheidungen und individuellen Konstruktionserfahrungen unabhängig zu machen. Dazu dienen besonders formale Beschreibungssprachen und strikte Klassifikationsprinzipien für die *exakte* Architekturdeskription und für eine automatische Exploration des "design space". Eine regelbasierte Konstruktionsmethodik mit vorab definierten Methodenschritten innerhalb eines voll durchsystematisierten Lösungsraumes soll hier den bisherigen "informal design process" weitestgehend ablösen (Dasgupta, 1984, Kap. 2). Der Entwicklungsgang von der Komplexität kunstvoll integrierenden Kompositionslehre zur Theorie-fundierten Dekompositionslehre schien damit zu seinem Abschluß gekommen zu sein.

Doch weder der Übergang zum konsequenten Parallelismus noch die endgültige Verwissenschaftlichung konnten sich bislang in der Computerarchitektur durchsetzen. Es traten vielmehr während der 90er Jahre Architek-turkonzepte in Erscheinung, die den konstruktiven, vortheoretischen Charakter des Rechnerdesigns wieder mehr in das Zentrum rücken. Auf der einen Seite kommt es ähnlich wie in der Softwarearchitektur zu einer Renaissance der Nutzer- bzw. Stakeholder-bezogenen Designorientierung und des deskriptiven Designraums (vgl. Shaw, Garlan 1996; Bass, Clements, Kazman 1998). Hier ragt neben dem "design space approach" von Sima, Fountain, Kacsuk (1997) besonders das große Werk von Blaauw und Brooks (1997) hervor, in dem die ästhetisch-funktionale Designqualität nun mit der Bell-Newellschen Designraum-Beschreibung verknüpft wird. Computerarchitektur versteht sich dabei wieder als systematisierte Bestandsaufnahme der historisch erkundeten Designalternativen, u. z. in der Absicht, das Gestaltungs- und Methodeninventar für optimal an die jeweiligen Verwendungszwecke angepaßte Rechnersysteme bereitzustellen.

Wandel der zentralen Ziele von Architekturkonzepten vor und nach Etablierung des Technikbereiches

Vor Etablierung: Erschließung und Ausweitung von Formenvielfalt und Designkomplexität

- Anpassung der Bauformen an Nutzerbedürfnisse
- Abstimmung der Designmerkmale / Zielkonflikte
- Abstimmung von Designer- und Auftraggeber / Betroffener
- Multiperspektivische Betrachtungs- und Vorgehensweise: Erschließung und Abstimmung der Perspektivenvielfalt

Variation der Struktur des Systems:

- Vielgestaltigkeit der Bauformen: Stile
- Erschließung aller Variationsmöglichkeiten durch umfassende Klassifikation der Bauformen
- Identifikation noch nicht realisierter Variations- und Konfigurationsmöglichkeiten
- Ausweitung des Designraumes
- Lösungsraum-Erschließung führt zur Mehrung von Designkomplexität

Seit Etablierung: Begrenzung der Formenvielfalt und Vermeidung von Designkomplexität

- Systematik der Bauformen: Ordnung der Gestaltungsvarianten durch mehrstufige Klassifikation
- Hierarchische Modellierung des gesamten Designraums zur Fixierung der Gestaltungsvarianten
- Vereinfachung des Entwurfs durch Kanalisierung der Varianten-Konfiguration
- Strukturierung des Konstruktionsgegenstandes und Entwurfsprozesses zur Erleichterung der Arbeitsteilung
- Rationalisierung des Entwurfsprozesses durch Verwissenschaftlichung / Algorithmisierung

Beherrschung und Reduktion der Designkomplexität durch Einfrieren der Struktur:

- zur Reduktion des Abstimmungsaufwandes,
- zur Reduktion der Variantenvielfalt,
- zur Sicherung der Kompatibilität,
- zur Bewahrung der Änderbarkeit
- Schließung, Abschottung der Lösungsrichtungen
- Standardisierung der Bauformen und Komponenten
- Standardisierung des Entwurfsprozesses

Auf der anderen Seite entstand, ausgelöst durch die RISC-Erfinder Hennessy und Patterson, eine sehr pragmatische Richtung, die sich auf konstruktive Engineeringmethoden zurückbesann. In diesem Ansatz wird zugunsten einer Markt- und Verwendungs-bezogenen Wirkungsgrad-Optimierung der gängigsten Bauformen weitgehend auf komplexe designwissenschaftliche Instrumentarien und Designraum-Taxonomien verzichtet. Stattdessen betrachtet man die Rechnerarchitektur wieder mehr als eine Art thermodynamisches Fließ- oder Gleichgewichtsmodell, das man mit Petrinetzen oder Warteschlangen quantitativ beschreiben kann. Nicht mehr die Vielfalt der Bauweisen oder ein Gesamtinventar der Bauformen sind hier das Ziel, sondern die Sondierung des Strategiespektrums und der Gestaltungsmittel, die die höchste Effizienzsteigerung der Bauweisen bewirken. Diese Richtung will mit der konsequent empirischen Fundierung und mit ausgefeilten ingenieurmäßigen Meß- und Testverfahren die bisher theorielastige Computerarchitektur entmystifizieren (Hennessy, Patterson 1990/95). Daß aber auch mit diesem quantifizierenden Architekturkonzept noch nicht der Abschluß der disziplinären Entwicklung erreicht ist, zeigen neuere Grundsatzüberlegungen Hennessys beim "Annual International Symposium on Computer Architecture" zur Jahrtausendwende: die Entwicklung zum "Embedded" und "Ubiquitous Computing" könnte danach eine Schwerpunktverlagerung von der in den letzten Jahrzehnten durch das Mooresche Gesetz angetriebenen leistungsorientierten quantitativen Architekturbetrachtung zu einem stärker qualitativen "focus for systems research" führen, bei dem die Verfügbarkeit, Sicherheit, Wartbarkeit und Skalierbarkeit im Zentrum stünden (Henessy 1999, S 27 ff.) Architekturverständnisse folgen damit aber, so zeigt sich einmal mehr, nicht vorwiegend einer internen disziplinären Logik, sondern erweisen sich als eminent abhängig von technisch-ökonomischen "Trajectories" (Nelson/ Winter) und gesellschaftlichen Prioritätensetzungen.

Diese Entwicklungstendenzen in der Computerarchitektur scheinen zu signalisieren, daß sich angesichts der Fortdauer der technischen Umwälzungen und der stets noch wachsenden Nutzungs- bzw. Umgebungs-bedingten Komplexität szientistische Methoden der Erfahrungskompression und Designrationalisierung nur begrenzt bewährt haben und daß sich die Disziplin damit wieder mehr als eine konstruktive Technikwissenschaft begreift. Deren Aufgabe besteht wesentlich in Allokationsentscheidungen bei Funktionen und knappen Betriebsmitteln, in der Organisation arbeitsteiliger Stukturen und Prozesse innerhalb der technischen Aggregate wie im Gesamtsystem sowie generell im Management

der divergierenden Nutzer- bzw. Stakeholder-Anforderungen in der Systemgestaltung. Und dieses Multiakteursspiel von Funktionsfindung, Ressour-cenallokation und Systemdimensionierung ergibt sich nicht aus noch so systematischen Bauformtaxonomien oder "algebraischen Systemtheorien" (Everling). Eine streng wissenschaftliche Modellierung nach Vorbild der Mathematik und Naturwissenschaften ist so nicht möglich, die Entwickler müssen hier wie in anderen designintensiven Ingenieurdisziplinen mit einer Vielfalt von Modellsichten und der Heterogenität von Designkonflikten zurecht-kommen. Architekturdesign bleibt so ein bewußter heterogener Gestaltungsakt, in dem 'rationale' Strukturbildung, hermeneutische Leitbilddiskurse und Verständigungprozesse sowie vor allem soziale Aushandlungsprozesse ineinandergreifen. Für dieses Technikfeld scheint so noch immer Brooks Feststellung von 1963 zu gelten: "In any advancing discipline, the systems actually employed are considerable more complex than those for which existing theory provides a complete analysis. Design, therefore, requires art as well as science, judgement as well as knowledge [...] So it is with data processing systems, there exists a small body of knowledge and techniques concerning the proper methods of combining equipment, human procedures, and computer programs into unified systems. There exists a larger body of experience from which have come few general rules but much advice" (Brooks, Iverson 1963, S. 437).

Überblickt man die Entwicklung der Wissenschaftskonzepte der Rechnerarchitektur, so sind bis zu einem gewissen Grade die von der Wissenschaftssoziologie und -geschichte herausgearbeiteten Theoriemodelle auch in diesem Technikfeld zu beobachten: Erfahrungsakkumulation, deskriptive Klassifikation und Systematisierung des Lösungsraumes, zunehmende Formalisierung und Quantifizierung der Methoden. Doch im Gegensatz zu den meisten Entwicklungsmodellen findet sich im Computer-bereich schon vor der empirischen Realisierung ein ungewöhnlicher Theorievorlauf, während der Übergang von der Erfahrungswissenschaft zu einer theoriebasierten 'exakten' ingenieurwissenschaftlichen Disziplin zwar am Ende der 50er Jahre und dann noch einmal in den 80er und 90er Jahren in Angriff genommen, aber bislang nicht abgeschlossen wurde. Im Gegenteil, frühere Verwissenschaftlichungs-Ansprüche wurden zugunsten einer Renaissance der Designorientierung und der Designraum-Deskription sowie empirischer Optimierungsstrategien wieder zurückgenommen. Da sich ähnliche Tendenzen auch in der Softwareentwicklung und sogar in der integrierten Produktentwicklung der Mechanikkonstruktion zeigen, liegt die Annahme nahe, daß es sich bei diesen ausgeprägt konstruktiven Technikdisziplinen um einen Typ von Technikwissenschaften

handelt, der sich prinzipiell von den mehr naturwissenschaftlich, technologischen oder grundlagenwissenschaftlich ausgerichteten Ingenieurdisziplinen unterscheidet und für den daher am Wissenschaftsverständnis der Physik und Mathematik orientierte Wissenschaftskonzepte nicht angemessen sind. Zemaneks Entwicklungsspirale bzw. seine These von der Re-Huma-ni-sierung der "abstrakten Architektur" erweist sich für den Computerbereich so als das adäquatere Modell als die linearen Modelle der Verwissenschaftlichung. Dies könnte auch der tiefere Grund dafür sein, daß die Architekturmetapher in der Computer Community bis heute ihr Feld behauptet und nun auch in der Software Community die bisherige Vorherrschaft des Engineering-Konzeptes gebrochen hat.

Bibliographie zur Geschichte der Informatik

ACM Curriculum Committee on Computer Science (Hrsg.), An Undergraduate Program in Computer Science. Preliminary Recommendations, in: Communications of the ACM 8 (1965), S. 543-552

ACM Curriculum Committee on Computer Science (Hrsg.), Curriculum 68. Recommendations for Academic Programs in Computer Science. A Report of the ACM Curriculum Committee in Computer Science, in: Communications of the ACM 11 (1968) 3, S. 151-197

Adam, Alison, What Can the History of AI Learn from the History of Science, in: AI & Society 4 (1990), S. 232-241

Adams, J. Mack; Moon, R., An Introduction to Computer Science, Glenview, Ill. 1970

Adams, J. Mack; Haden, Douglas H., Computers: Appreciation, Applications, Implications. An Introduction, New York, London, Sydney, Toronto 1973

Aiken, Howard H., The Future of Automatic Computing Machinery, in: Hoffmann, Walter; Walther, Alwin (Hrsg.), Elektronische Rechenmaschinen und Informationsverarbeitung (Nachrichtentechnische Fachberichte, Bd. 4), Braunschweig 1956, S. 31-35

Amarel, Saul, Computer Science: A Conceptual Framework for Curriculum Planning, in: Communications of the ACM 14 (1971) 6, S.

Amarel, Saul, Computer Science, in: Ralston, Anthony; Reilly, Jr., Edwin D. (Hrsg.), Encyclopedia of Computer Science and Engineering, New York 1976, S. 314-318

Arden, Bruce W. (Hrsg.), What Can Be Automated?: The Computer Science and Engineering Research Study (COSERS) Cambridge, MA 1980

Arsac, Jacques, La science informatique, Paris 1970

Aspray, William, From Mathematical Constructivity to Computer Science: Alan Turing, John von Neumann, and the Origins of Computer Science and Mathematical Logic, Phil. Diss. University of Wisconsin, Madison 1980

Aspray, William, The Scientific Conceptualization of Information, in: Annals of the History of Computing, 7 (1985) 2, S. 117-140

Aspray, William, John von Neumann's Contributions to Computing and Computer Science, Annals of the History of Computing. 11 (1989) 3, S. 189-195

Aspray, William, The Origins of John von Neumann's Theory of Automata, in: The Legacy of John von Neumann, Providence, R.I., 1990, S. 289-309

Aspray, William, Was Early Entry a Competitive Advantage? US Universities that Entered Computing in the 1940s, in: Annals of the History of Computing 22 (2000) 3, S. 42-87

Atchison, William F.; Hamblen, John W., Status of Computer Science in Colleges and Universities, in: Communications of the ACM 7 (1964) 4, S. 225-227

Bajcsy, R.; Reynolds, C., Computer Science: The Science of and about Information and Computation, in: Communications of the ACM 45 (2002) 3, S. 94-98

Bauer, Friedrich L., Der Computer in unserer Welt. Festrede, gehalten in der Bayerischen Akademie der Wissenschaften in München am 6. Dezember 1969, München 1970

Bauer, Friedrich L., Was heißt und was ist Informatik? Merkmale zur Orientierung über eine neue wissenschaftliche Disziplin, in: IBM Nachrichten Nr. 223 (1974), S. 333-337

Bauer, Friedrich L., Was heißt und was ist Informatik? in: Otte, M. (Hrsg.), Mathematiker über die Mathematik, Berlin, Heidelberg, New York 1974, S. 349-368

Bauer, Friedrich L., Mathematik und Informatik, in: Dörfler, Willibald; Schauer, Helmut (Hrsg.), Wechselwirkungen zwischen Informatik und Mathematik, Wien, München 1980, S. 17-39

Bauer, Friedrich L., Kurzer Abriß der Geschichte der Informatik 1890-1990, in: Fischer, G. u.a. (Hrsg.), Ein Jahrhundert Mathematik 1890-1990. Festschrift zum Jubiläum der DMV, Braunschweig, Wiesbaden 1990, S. 113-147

Bauer, Friedrich L., Informatik und Algebra, in: Broy, Manfred (Hrsg.), Informatik und Mathematik, Berlin 1991, S. 28-42

Bauer, Friedrich L., The Might of Formulas and Their Limits, in: Freksa, Christian; Jantzen, Mattias; Valk, Rüdiger (Hrsg.), Foundations of Computer Science. Potential, Theory, Cognition, (Lecture Notes in Computer Science, Bd. 1337), Berlin, Heidelberg, New York 1997, S. 1-8

Bauer, Friedrich L.; Goos, Gerhard, Informatik: Eine einführende Übersicht, 2 Bde., 1. Aufl. Berlin, Heidelberg, New York 1970/71 ; 2. Aufl. Berlin, Heidelberg, New York 1973/74 ; 3. Aufl. Berlin, Heidelberg, New York 1982/84; 4. Aufl. Berlin, Heidelberg, New York 1990

Bauer, Friedrich L.; Heinold, Josef; Samelson, Klaus; Sauer, Robert, Moderne Rechenanlagen (Leitfäden der angewandten Mathematik und Mechanik, Bd. 5), Stuttgart 1965

Bauer, Friedrich L.; Weinhart, K., Informatik - eine grundlegende Einführung, München 1974

Bemer, Robert W., Über den Computer in unserer Gesellschaft, in: Elektronische Rechenanlagen 19 (1977) 4, S. 167-172

Biermann, Alan W., Great Ideas in Computer Science: a Gentle Introduction, Cambridge u.a. 1990

Bolkart, W.; Haug, H., Die Utopie des beherrschten Mediums, in: Computer Magazin 5/1986, S. 14-16 (über Carl Adam Petri)

Bonsiepen, Lena; Coy, Wolfgang, Eine Curriculardebatte, in: Informatik Spektrum 15 (1992) 6, S. 323-325

Bonsiepen, Lena; Coy, Wolfgang, Softwareerstellung zwischen formalen Methoden und arbeitsorientierter Gestaltung - Eine Curriculardebatte, in: Wolfgang Coy, Peter Gorny, Ilona Kopp, Constantin Skarpelis: Menschengerechte Software als Wettbewerbsfaktor, Arbeitstagung des Projektträgers "Arbeit und Technik" in Zusammenarbeit mit dem German Chapter of the ACM und der Gesellschaft für Informatik am 27. und 28. Januar 1993 in Bonn, Stuttgart 1993, S. 131-139

Brauer, Wilfried, The New Paradigm of Informatics, in: Maurer, H. (Hrsg.), New Results and New Trends in Computer Science (Lecture Notes in Computer Science, Bd. 555), Berlin, Heidelberg, New York 1991, S. 15-24

Brauer, Wilfried, Betrachtungen der Informatik, in: Informations- und Kommunikationstechnik (IuK) als fachübergreifende Aufgabe, Augsburger Universitätsreden, Heft 38, Augsbug 1999, S. 30-43

Brauer, Wilfried, Mathematik und Informatik, in: Mitteilungen der Mathematischen Gesellschaft Hamburg 19 (2000), S. 55-67

Brauer, Wilfried, Informatikbetrachtungen. Versuch einer Beschreibung des Fachs Informatik, in: Desel, Jörg (Hrsg.), Das ist Informatik, Berlin, Heidelberg, New York 2001, S. 23-32

Brauer, Wilfried; Brauer, Ute, Informatik - das neue Paradigma. Änderungen von Forschungszielen und Denkgewohnheiten der Informatik. In: LOGIN 15 (1995) 4, S. 25-29

Brauer, Wilfried; Brauer, Ute, Wissenschaftliche Herausforderungen für die Informatik: Änderungen von Forschungszielen und Denkgewohnheiten, in: Langenheder, W.; Müller, G.; Schinzel, B., Informatik cui bono? GI-FB 8 Tagung Freiburg, 23.- 26. September 1992, Berlin, Heidelberg, New York, S. 11-19

Brauer, Wilfried; Haake, Wolfhart; Münch, Siegfried, Studien- und Forschungsführer Informatik, 1. Aufl. Berlin, Heidelberg, New York 1982, 2. Aufl. 1989

Brauer, Wilfried; Haake, Wolfhart; Münch, Siegfried, Studien- und Forschungsführer Informatik, Bonn: GMD und Bad Godesberg: DAAD, 1. Aufl. 1973, 2. Aufl. 1974, 3. Aufl. 1978, 4. Aufl. 1980

Brauer, Wilfried; Münch, Siegfried, Studien- und Forschungsführer Informatik, 3. Aufl. Berlin, Heidelberg, New York 1996

Breton, Philippe, Histoire de l'informatique, Paris 1987, 2.Aufl. 1990

Brooks Jr., Frederick P., The Computer "Scientist" as Toolsmith - Studies in Interactive Computer Graphics, in: Information Processing 1977, Proceedings of IFIP Congress '77, Amsterdam, London 1977, S. 625-634

Broy, Manfred, Informatik. Eine grundlegende Einführung, Bd. 1: Programmierung und Rechnerstrukturen, 2. Aufl. Berlin, Heidelberg, New York 1998

Broy, Manfred; Schmidt, Joachim W., Informatik: Grundlagenwissenschaft oder Ingenieurdisziplin? in: Informatik Spektrum 22 (1999), S. 206-209

Büttemeyer, Wilhelm, Wissenschaftstheorie für Informatiker, Heidelberg, Berlin, Oxford 1995

Bundesminister für wissenschaftliche Forschung, Empfehlungen zur Ausbildung auf dem Gebiet der Datenverarbeitung, in: Internationale Elektronische Rundschau 8 (1968), S.211

Carberry, M. S.; Khalil, H. M.; Leathrum, J. F.; Levy, L. S., General Computer Science, Columbus, Ohio 1976

Cardenas, Alfonso F., Presser, Leon; Marin, Miguel A. (Hrsg.), Computer Science, New York, London, Sydney 1972

Claus, Volker, Einführung in die Informatik, Stuttgart 1975

Claus, Volker, Was ist Informatik und Abgrenzungen und Abgrenzungen zur Mathematik, in: Dörfler, Willibald; Schauer, Helmut (Hrsg.), Wechselwirkungen zwischen Informatik und Mathematik, Wien, München 1980, S. 40-82

Claus, Volker, Stellenwert theoretischer Inhalte in universitären Informatikstudiengängen., in: GI Jahrestagung 1995, Huber-Wäschle, Friedbert; Schauer, Helmut; Widmayer, Peter (Hrsg.), Herausforderungen eines globalen Informationsverbundes für die Informatik,

25. GI-Jahrestagung und 13. Schweizer Informatikertag, GISI 95, Zürich, 18.-20. September 1995, Berlin, Heidelberg, New York 1995, S. 391-398

Claus, Volker, Informatik und Ausbildung, in: GI-Fachtagung 98, Informatik und Ausbildung, Stuttgart, 30. März - 1. April 1998, Berlin, Heidelberg, New York 1998

Coy, Wolfgang, Of Mice and Maze. Elektronische Informationsverarbeitung und Kybernetik 14 (1978) 5, S. 227-232

Coy, Wolfgang, Brauchen wir eine Theorie der Informatik? in: Informatik-Spektrum, 12 (1989) 5, S. 256-266

Coy, Wolfgang, Informatik – Eine Disziplin im Umbruch, in: ders. u.a. (Hrsg.), Sichtweisen der Informatik, Braunschweig, Wiesbaden 1992, S. 1-9

Coy, Wolfgang, Für eine Theorie der Informatik, in: ders. u.a. (Hrsg.), Sichtweisen der Informatik, Braunschweig, Wiesbaden 1992, S. 17-32

Coy, Wolfgang, Reduziertes Denken. Informatik in der Tradition des formalistischen Forschungsprogramms, in: Informatik und Philosophie 1992, S. 31-52

Coy, Wolfgang, Der moderne Charakter des Computers, in: Kreowski, Hans-Jörg (Hrsg.): Informatik zwischen Wissenschaft und Gesellschaft. Zur Erinnerung an Reinhold Franck (Informatik-Fachberichte, Bd. 309), Berlin, Heidelberg, New York 1992, S. 255-265

Coy, Wolfgang, Cultural Stability and Technological Change: The Case of Information, Communication and Media Technology., in: IFIP Congress (3) 1994, S. 210-217

Coy, Wolfgang, Der blinde Fleck der Wissenschaft, in: Hans-Jörg Kreowski, Thomas Risse, Andreas Spillner, Ralf E. Streibl, Karin Vosseberg (Hrsg.), Realität und Utopien der Informatik (ausgewählte Beiträge der 10. Jahrestagung des Forums Informatikerinnen und Informatiker für Frieden und gesellschaftliche Verantwortung e.V., 7.-9. Oktober 1994, Bremen) agenda Verlag 1995, S. 215-217

Coy, Wolfgang, Automat - Werkzeug - Medium, in: Informatik Spektrum 18 (1995) 1, S. 31 – 38

Coy, Wolfgang, Informatik und Gesellschaft, in: Wilhelm, Reinhard (Hrsg.), Informatik – Grundlagen, Anwendungen, Perspektiven, München 1996, S. 125-132

Coy, Wolfgang, Was ist, was kann, was soll „Informatik und Gesellschaft“? in: Schinzel, Britta (Hrsg.), Schnittstellen – Zum Verhältnis von Informatik und Gesellschaft, Braunschweig, Wiesbaden 1996, S. 17-27

Coy, Wolfgang, Defining Discipline, in: Freksa, Christian; Jantzen, Matthias; Valk, Rüdiger (Hrsg.), Foundations of Computer Science. Potential, Theory, Cognition, (Lecture Notes in Computer Science, Bd. 1337), Berlin, Heidelberg, New York 1997, S. 21-35

Coy, Wolfgang, Was ist Informatik? Zur Entstehung und zum Selbstverständnis des Faches an den deutschen Universitäten, in: Desel, Jörg (Hrsg.), Das ist Informatik, Berlin, Heidelberg, New York 2001, S. 1-22

Coy, Wolfgang, Was ist Informatik? Zur Entstehung und zum Selbstverständnis des Faches an den deutschen Universitäten, erscheint in: Hellige, Hans Dieter (Hrsg.), Geschichten der Informatik. Visionen, Paradigmen und Leitmotive, Berlin, Heidelberg, New York 2003

Coy, Wolfgang; Nake, Frieder; Pflüger, Jörg-Martin; Rolf, Arno; Siefkes, Dirk; Seetzen, Jürgen; Stransfeld, Reinhard (Hrsg.), Sichtweisen der Informatik, Braunschweig, Wiesbaden 1992

Claus, Volker; Wilhelm, Reinhard, Einleitung., in: Wilhelm, Reinhard (Hrsg.), Informatik – Grundlagen, Anwendungen, Perspektiven, München 1996, S. 9-12

Denning, Peter J., Computing, Applications, and Computational Science, in: Communications of the ACM 34 (1991) 10, S. 129-131

Denning, Peter J., What is Computer Science?, in: American Scientist 73 (Jan./Feb. 1985), S. 16-19

Denning, Peter J., A Debate on Teaching Computing Science, in: Communications of the ACM 32 (1989) 12, S. 1397-1414

Denning, Peter J., Beyond Formalism, in: American Scientist 79 (Jan./Feb. 1991), 1991, S. 8-10

Denning, Peter J., Educating a New Engineer, in: Communications of the ACM 35 (1992) 12, S. 82-97

Denning, Peter J., Can There Be a Science of Information? in: Computing Surveys 27 (1995) 1, S. 23-25

Denning, Peter J., Who Are We? in: Communications of the ACM 44 (2001) 2, S. 15-19

Denning, Peter J.; Comer, Douglas E.; Gries, David u.a., Computing as a Discipline, in: Communication of the ACM 32 (1989) 1, S. 9-23

Denning, Peter J.; Feigenbaum, Edward A.; Gilmore, Paul; Hearn, Anthony C.; Ritchie, Robert W.; Traub, J. F., A Discipline in Crisis, in: Communication of the ACM 24 (1981) 6, S. 370-374

Dennis, Jack B., Interrelating hardware and software in computer science education, in: AFIPS, Bd. 34, Spring Joint Computer Conference 1969, S. 537-538
Desel, Jörg (Hrsg.), Das ist Informatik, Berlin, Heidelberg, New York 2001
Devaux, P., Automates et automatismes, Paris 1941
Devaux, P., Automates, Automatismes, Automation, Paris 1967
Dijkstra, Edsger W., Programming Considered as a Human Activity, in: Information Processing 1965, Proceedings of IFIP Congress '65, 2 Bde. Washington, D. C., London 1965, Bd. , S. 213-217
Dijkstra, Edsger W., A Discipline of Programming, Englewood Cliffs, N. J. 1976
Dijkstra, Edsger W., The Cruelty of Really Teaching Computing Science, in: Communications of the ACM, 32 (1989) 12, S. 1398 - 1414
Dörfler, Willibald; Schauer, Helmut (Hrsg.), Wechselwirkungen zwischen Informatik und Mathematik, Wien, München 1980
Donth, Hans H., Der Aufbau der Informatik an Deutschen Hochschulen, in: Elektronische Rechenanlagen 26 (1984) 5, S. 223-228
Dorf, Richard C., Introduction to Computers and Computer Science, San Francisco 1972; 2. Aufl.
Endres, Albert, Die Informatik als Ingenieurwissenschaft, in: Informatik Spektrum, 22 (1999), S. 439-443
Engelhardt, Dietrich v. (1978), Diskussionsbeitrag „Dimensionen und Aspekte der Entstehung neuer Wissenschaften in der Neuzeit, in: Beiträge zur Wissenschaftsgeschichte 1 (1987), S. 173-174
Engesser, Hermann; Claus, Volker; Schwill, Andreas (Hrsg.), Duden Informatik: ein Sachlexikon für Studium und Praxis, Mannheim u.a. 2. Aufl. 1993
Eulenhöfer, Peter, Der Informatiker als "deus ex mathematica", in: Siefkes, Dirk; Eulenhöfer, Peter; Stach, Heike; Städtler, Klaus (Hrsg.), Sozialgeschichte der Informatik. Kulturelle Praktiken und Orientierungen, Wiesbaden 1998, S. 257-273
Eulenhöfer, Peter, Die formale Orientierung der Informatik. Zur mathematischen Tradition der Disziplin in der Bundesrepublik Deutschland. Diss., FB Informatik, TU Berlin 1999
Fakultätentag Informatik & Fakultätentag Elektrotechnik, Gemeinsame Erklärung zur Informationstechnik, 1991
Fein, L., The Role of the University in Computers, Data Processing and Related Fields, in: Communications of the ACM 2 (1959) 9, S. 7-11
Finerman, Aaron (Hrsg.), University Education in Computing Science, New York, London 1968
Forsythe, Alexandra I.; Keenan, T.; Organick, Elliott I.; Stenberg, W., Computer Science: A First Course, New York u.a. 1969 (15a kyb 401 e/075)
Forsythe, George E., University Education in Computing Science, in: AFIPS Bd. 26, Fall Joint Computer Conference, Nov 1964, S.
Forsythe, George E., A University's Education Program in Computing Science, in: Communications of the ACM 10 (1967) 1, S. 3-11
Frank, Ulrich, Informatik und Wirtschaftsinformatik. Grenzziehungen und Ansätze zur gegenseitigen Befruchtung, in: Desel, Jörg (Hrsg.), Das ist Informatik, Berlin, Heidelberg, New York 2001, S. 47-66
Freksa, Christian; Jantzen, Mattias; Valk, Rüdiger (Hrsg.), Foundations of Computer Science. Potential, Theory, Cognition, (Lecture Notes in Computer Science, Bd. 1337), Berlin, Heidelberg, New York 1997
Friedewald, M. (1999), Der Computer als Werkzeug und Medium. Die geistigen und technischen Wurzeln des Personalcomputers (Aachener Beiträge zur Wissenschafts- und Technikgeschichte des 20. Jahrhunderts, Bd. 3) Berlin, Diepholz
Friedrich, Jürgen; Entwicklungslinien der Informatik, in: WSI-Mitteilungen 12/1988, S. 678-686
Friedrich, Jürgen; Herrmann, Thomas; Peschek, Max; Rolf, Arno (Hrsg.), Informatik und Gesellschaft, Heidelberg, Berlin, Oxford 1995
Ganzhorn, Karl E., Beziehungen zwischen Informationsverarbeitung und Kommunikation, in: Endres, Albert; Schünemann, Claus (Hrsg.), Informationsverarbeitung und Kommunikation. Informatiksymposium 1978 der IBM Deutschland, Bad Neuenahr, Sept. 1978, München, Wien 1979, S. 9-23
Ganzhorn, Karl E., Schwellen des Fortschritts in der Informationstechnik, in: Methner, H.; Oberhoff, W. D.; Schmidt, P.; Swiridow, A. P.; Unger, H.; Vollmar, R. (Hrsg.), Computer und Kybernetik. Anmerkungen zu ihrer Geschichte und zu Perspektiven in der Zeit von 1940 bis 1965. 3. Russisch-Deutsches Symposium, Heidelberg November 1997; S. 45-53

Genrich, H. J., Belästigung der Menschen durch Computer, in: GI – 5. Jahrestagung, Dortmund 1975, Berlin, Heidelberg, New York 1975, S. 94-105; und in: GMD-Spiegel 1975, Nr. 5, S. 32-45

Giloi, Wolfgang K., Was ist Informatik?, in: TUB 2 (1970) 1, S. 4-15

Goldreich, O.; Widgerson, A., Theory of Computing: A Scientific Perspective, unter der URL: http://theory.lcs.mit.edu/oded/toc-sp.html, 1996

Gorn, Saul, The Computer and Information Science: A New Basis Discipline, in Society for Industrial and Applied Mathematics (SIAM) 5 (1963), S. 150-155

Graf, Klaus-Dieter, Informatik. Eine Einführung in Grundlagen und Methoden, Freiburg, Basel, Wien 1981

Gries, David, Teaching Calculation and Discrimination: A more Effective Curriculum, in: Communications of the ACM 34 (1991) 3, S. 44-55

Gries, David; Personius (Hrsg.), Cornell Computer Science Vision for the Next Decade. Annual Report Education and Research 1996-1997, Department of Computer Science, Cornell University, 1998, (http://www.cs.cornell.edu/annual_report/96-97/vision.htm

Gruenberger, Fred J., Computing: An Introduction, New York 1969

Haacke, W.; Fischbach, F., Informatik, Bad Honnef 1972

Hagen, Wolfgang, Computerpolitik, , in: Bolz, Norbert, Kittler, Friedrich A.; Tholen, Christoph (Hrsg.), Computer als Medium, München 1994, S. 139-160

Hamming, Richard W., One Man's View of Computer Science (1968 ACM Turing Lecture), in: Journal of the ACM, 16 (1969) 1, S. 3-12

Hamming, Richard W., Computers and Society, New York u.a. 1972

Hartmann, B., Die elektronische Datenverarbeitung in Lehre und Praxis. Ergebnisse einer Studienreise nach den USA, Berlin, Köln, Frankfurt a. M. 1959

Hartmannis, J.; Lin, H. (Hrsg.), Computing the Future: A Broader Agenda for Computer Science and Engineering, Washington, D.C.; Zusammenfassung in: in: Communications of the ACM, 35 (1992) 11, S.

Hashagen, Ulf ; Keil-Slawik, Reinhard; Norberg, Arthur L. (Hrsg.), History of Computing :Software Issues: International Conference on the History of Computing, Ichc 2000, April 5-7, 2000, Heinz Nixdorf Museums Forum, Paderborn, Germany, Berlin, Heidelberg, New York 2002

Heintz, Bettina, Die Herrschaft der Regel. Zur Grundlagengeschichte des Computers, Frankfurt, New York 1993

Helling, Klaus, Informatiker – Beruf ohne Berufsbild? in: Informatik Spektrum, 16 (1993) S. 222-224

Hellige, Hans Dieter, (Hrsg.), Leitbilder der Informatik- und Computer-Entwicklung. Eine Tagung der Fachgruppe "Historische Aspekte von Informatik und Gesellschaft" der Gesellschaft für Informatik und des Deutschen Museums, München 4.-6.Oktober 1993, Forschungszentrum Arbeit und Technik, Universität Bremen, artec-Paper Nr. 33, Dezember 1994, 478 S.

Hellige, Hans Dieter, (Hrsg.), Technikleitbilder auf dem Prüfstand. Das Leitbild-Assessment aus Sicht der Informatik- und Computergeschichte“, Berlin 1996

Hellige, Hans Dieter (Hrsg.), Geschichten der Informatik. Visionen, Paradigmen und Leitmotive, Berlin, Heidelberg, New York erscheint 2003

Heyderhoff, P.; Hildebrand, Th., Informationsstrukturen. Eine Einführung in die Informatik, Mannheim, Wien, Zürich 1973

Hoare, Charles Antony Richard, Computer Science, New Lecture Series 62, Queen’s University, Belfast 1971; wiedergedr. in: ders.: Essays in Computer Science, hrsg. von Jones, Cliff B., New York, London, Toronto 1989, S. 89-101

Hoare, Charles Antony Richard, The Emperor's Old Clothes (The 1980 Turing Award Lecture), in: Communications of the ACM, 24, (1981) 2, S. 75-83; wiedergedr. in: ders.: Essays in Computer Science, hrsg. von Jones, Cliff B., New York, London, Toronto 1989, S. 1-18; als Übersetzung: Der neue Turmbau zu Babel. Rede zur Verleihung des Turing-Preises der Association for Computing Machinery, in: Kursbuch 75, Berlin 1980, S. 57-73

Hohn, Hans-Willy, Kognitive Strukturen und Steuerungsprobleme der Forschung: Kernphysik und Informatik im Vergleich (Schriften des Max-Planck-Instituts für Gesellschaftsforschung Köln, Bd. 36), Frankfurt a. M., New York 1998

Hopcroft, John E., Computer Science: The Emergence of a Discipline, in: Communications of the ACM 30 (1987) 3, S. 198-202

Hotz, Günter, Einführung in die Informatik, Stuttgart 1990

Hubwieser, Peter; Broy, Manfred, Grundlegende Konzepte von Informations- und Kommunikationssystemen für den Informatikunterricht, in: INFOS 1997, S. 40-50

Ivall, T. E., 50 Years of Computer Science, in: Electronics and Wireless World 92 (1986), S. 52-62

Jones, Anita K., Perspectives on Computer Science. From the 10th Anniversary Symposium at the Computer Science Department, Carnegie-Mellon University, New York, San Francisco, London 1977 (A kyb 3000 tp 941)

Keenan Thomas A., Computers and Education, in: Communications of the ACM 7 (1964) 4, S. 214-215

Klaeren, H., Vom Problem zum Programm. Eine Einführung in die Informatik, Stuttgart 1990

Krabbel, Anita; Kuhlmann, Bettina; Brunnstein, Klaus; Oberquelle, Horst (Hrsg.), Zur Selbstverständnis-Diskussion der Informatik, Fachbereich Informatik, Universität Hamburg, Bericht FBI-HH-B- 169/94, Hamburg 1994

Krämer, Sybille, Symbolische Maschinen. Die Idee der Formalisierung in geschichtlichem Abriss, Darmstadt 1988

Krämer, Sybille, Symbolische Maschinen, Computer und der Verlust des Ethischen im geistigen Tun, in: Coy, Wolfgang u. a. (Hrsg), Sichtweisen der Informatik, Wiesbaden 1992, S. 335-341

Laplante, Phillip (Hrsg.), Great Papers in Computer Science, St. Paul, Minn. 199X

Lehmann, N. Joachim, Alwin Walther und seine Mathesis, in: Technische Universität Darmstadt (Hrsg.), Alwin Walther: Pionier des Wissenschaftlichen Rechnens. Wissenschaftliches Kolloquium anläßlich des hundertsten Geburtstages, Darmstadt Mai 1998 (TUD Schriftenreihe Bd. 75), Darmstadt 1999, S. 15-36

Ligonnière, R., Les origines de l'informatique anglaise, in: L'Informatique professionelles 24 / Juni/Juli 1984; 26 / Oktober 1984; 27 / November 1984

Lilley, Dorothy B.; Trice R. W., A History of Information Science 1945-1985, San Diego 1989

Lindner, Rudolf; Wohak, Bertram; Zeltwanger, Holger, Planen, Entscheiden, Herrschen. Vom Rechnen zur elektronischen Datenverarbeitung (Deutsches Museum, Kulturgeschichte der Naturwissenschaften und der Technik), Reinbek 1984

Luft, Alfred Lothar, Informatik als Technikwissenschaft, Mannheim, Wien, Zürich 1988

Luft, Alfred Lothar, „Wissen" und „Information" bei einer Sichtweise der Informatik als Wissenstechnik, in: Coy, Wolfgang u.a. (Hrsg.), Sichtweisen der Informatik, Braunschweig, Wiesbaden 1992, S. 49-70

Luft, Alfred Lothar; Kötter, Rudolf, Informatik - Eine moderne Wissenstechnik, Mannheim Wien, Zürich 1994

Mahoney, Michael, Computers and Mathematics: The Search for a Discipline of Computer Science, in: Echeverría, Javier; Ibarra, A ; Mormann, T. (Hrsg.),The Space of Mathematics, Berlin, New York, 1992, S. 347-61

Mahoney, Michael, Computer Science: The Search for a Mathematical Theory, in: Krige, John; Pestre, Dominique (Hrsg.), Science in the 20th Century, Amsterdam 1997, Kap. 31

Mahoney, Michael, The Structures of Computation, in: Rojas, Raúl; Hashagen, Ulf (Hrsg.), The First Computers - History and Architectures, Cambridge, MA, London 2000, S. 17-32

Mahr, Bernd, Informatik. Wachstumsstörungen eines Wunderkinds, in: Kursbuch H. 97 (1989), S. 55-70

Mainzer, Klaus, Entwicklungsfaktoren der Informatik in der Bundesrepublik Deutschland, in: Daele, Wolfgang van den; Krohn, Wolfgang; Weingart, Peter (Hrsg.), Geplante Forschung, Frankfurt a. M. 1979, S. 117-180

Mainzer, Klaus, Computer - Neue Flügel des Geistes? Die Evolution computergestützter Technik, Wissenschaft, Kultur und Philosophie, Berlin 1994

Maisel, H., Introduction to Electronic Computers, New York u.a. 1969

Mallet, R. A., La méthode informatique. Conception et réalisation de l'informatique de gestion, Paris 1971

Martin, James; Bernstein, Philip A.; Denning, Peter J.; Dertouzos, Michael L.; Kleinrock, Leonard, Computer Science Education Today, A Dialogue. Communications of the ACM 28 (1985) 3, S. 251-262

Mayr, H. C; Maas, J., Perspektiven der Informatik, in: Informatik Spektrum, 25 (2002), S. 177-186

Methner, H.; Oberhoff, W. D.; Schmidt, P.; Swiridow, A. P.; Unger, H.; Vollmar, R. (Hrsg.), Computer und Kybernetik. Anmerkungen zu ihrer Geschichte und zu Perspektiven in der Zeit von 1940 bis 1965. 3. Russisch-Deutsches Symposium, Heidelberg November 1997

Meyer, Albert R.; Guttag, John V.u.a. (Hrsg.), Research Directions in Computer Science, Cambridge, Mass., London 1991

Möhring, Manfred, Zur Frühgeschichte der Informatik in der DDR, 1. Ribnitzer Informatik-historisches Kolloquium (Rostocker Wissenschaftshistorische Manuskripte, H. 19), Universität Rostock, Sektion Geschichte 1990, S. 18-31

Moreau, René, Ainsi naquit l'informatique, Paris 1981

Mounier-Kuhn, Pierre-E., L'Informatique en France depuis 1945: histoire d'une politique scientifique et industrielle, Thèse Paris, Conservatoire nationale des Arts et des Mètiers, Paris 1995

Mulder, Michael C.; Denning, Peter J., Computing in the Frontiers of Science and Engineering - Introduction, in: IEEE Computer 18 (1985) 11, S. 10-11

Nake, Frieder, Informatik und die Maschinisierung von Kopfarbeit, in: Coy, Wolfgang u.a. (Hrsg.), Sichtweisen der Informatik, Braunschweig, Wiesbaden 1992, S. 181-201

Nake, Frieder (Hrsg.), Die erträgliche Leichtigkeit der Zeichen. Ästhetik, Semiotik, Informatik, Baden-Baden 1993

Nake, Frieder, Der semiotische Charakter der informatischen Gegenstände, in: Bayer, U.; Gfesser, K.; Hansen, J. (Hrsg.), Signum um Signum. Elisabeth Walther-Bense zu Ehren, Semiosis Heft 85-90, (1997), S. 24-35

Nake, Frieder, Schwierigkeiten beim semiotischen Blick auf die Informationsgesellschaft, in: Zimmermann, Harald u.a. (Hrsg.), Knowledge Management und Kommunikations-systeme, Procedings 6. Internationales Symposium für Informationswissenschaft, Prag November 1998, Konstanz 1998, S. 455-468

Nake, Frieder, Was heißt und zu welchem Ende studiert man Informatik? Ein akademischer Diskursbeitrag nebst Anwendung, in: Claus, Volker (Hrsg.), Informatik und Ausbildung, Berlin, Heidelberg, New York, 1998, S. 1-13

Nake, Frieder, Das algorithmische Zeichen, in: Bauknecht, W.; Brauer, B.; Mück, Th. (Hrsg.), Informatik 2000. Tagungsband der GI/OCG Jahrestagung 2001, Bd. II, S. 736-742

Nake, Frieder, Begegnung im Zeichen. Informatik, Medium, Design, in: van den Boom, Holger (Hrsg.), Entwerfen, Jahrbuch 4 der HBK Braunschweig, Köln 2000, S. 174-186

Nake, Frieder, Kalkulierte und kalkulierende Zeichen. Der Computer als instrumentales Medium, in: Demuth, Volker; Wagner, Robin (Hrsg.), Vom Sinn multipler Welten. Medien und Kunst, Würzburg 2000, S. 121-140

Nake, Frieder, Arno Rolf, Dirk Siefkes (Hrsg.) Informatik - Aufregung zu einer Disziplin. Tagung Heppenheim 2001. Universität Hamburg, FB Informatik, Bericht 235, Hamburg 2001

Naumann, Friedrich, Die Genese der Informationsverarbeitung als technikwissenschaftliche Disziplin, Habilitationsschrift, Fakultät für Gesellschaftswissenschaften, TU Dresden 1984

Naumann, Friedrich, Informatik - neue Möglichkeiten der Datenverarbeitung, in: Buchheim, Gisela; Sonnemann, Rolf (Hrsg.), Geschichte der Technikwissenschaften, Leipzig 1990, S. 468-482

Naumann, Friedrich, Vom Abakus zum Internet. Geschichte der Informatik, Berlin 2001

Naur, Peter, Computing and the So-Called Foundations of the So-Called Sciences (1990), in: Peter Naur, Computing: A Human Activity, New York 1992, S. 49-63

Naur, Peter, Computing: A Human Activity, New York, Reading, Mass. 1992

Newell, Alan; Perlis, Alan; Simon, Herbert A., What is Computer Science?, in: Science 157 (1967), S. 1373-1374; wiedergedruckt in: Abacus 4 (1987) 4, S 32-

Newell, Alan; Simon, Herbert A., Computer Science as Empirical Inquiry, in: Communications of the ACM 19 (1976), S. 113-26

Nickel, Karl, Informatik – eine neue Wissenschaft. Interner Bericht 69/12 des Instituts für Informatik, Universität Karlsruhe, 1969; wiedergedruckt in: fridericiana. Zeitschrift der Universität Karlsruhe (1970) 6, S. 23-38

Nievergelt, Jürg, Welchen Wert haben theoretische Grundlagen für die Berufspraxis? Gedanken zum Fundament des Informatik-Turms, in: Informatik Spektrum, 18 (1995) 6, S. 342-344

Oberliesen, Rolf, Informationen, Daten und Signale, Hamburg 1982

Parnas, David L., Education for Computing Professionals, in: IEEE Computer 23 (1990) 1, S. 17-22

Perlis, Alan J., The Computer and the University, in: Greenberger, Martin (Hrsg.), Manage-ment and the Computer of the Future, Cambridge, Mass., New York, London 1962, S.181-199 (Panel und General Discussion, S. 199-217); Paperback-Edition unter dem Titel: Computers and the World of the Future, Cambridge, Mass., London 1962

Perlis, Alan J., Computer Science is neither Mathematics nor Electrical Engineering, in: A. Finerman (Hg.), University Education in Computing Science, New York, London 1968, S. 69-77

Perlis, Alan J., Identifying and developing curricula in software engineering, in: AFIPS, Bd. 34, Spring Joint Computer Conference 1969, S. 540-541
Perlis, Alan J., Introduction to Computer Science, New York 1972
Perlis, Alan J., Current Research Frontiers in Computer Science, in: Dertouzos, Michael L.; Moses, Joel (Hrsg.), Future Impact of Computers: A Twenty Year View, Cambridge, Mass. 1979, S. 422-436
Petri, Carl Adam, Kommunikationsdisziplinen. Gesellschaft für Mathematik und Datenverarbeitung (Bonn), Interner Bericht, Bonn, 30. März 1976
Petri, Carl Adam (Hrsg.), Ansätze zur Organisationstheorie rechnergestützter Informationssysteme, Fachtagung der Gesellschaft fuer Informatik (GI). Gesellschaft für Mathematik und Datenverarbeitung 1974, (Berichte der Gesellschaft für Mathematik und Datenverarbeitung ; 111), München, Wien 1979 (A kyb 550 i/620; TB BHV:kyb 550/3)
Petri, Carl Adam (Hrsg.), Kommunikationsdisziplinen, in: ders. (Hrsg.), Ansätze zur Organisationstheorie rechnergestützter Informationssysteme, Berichte der Gesellschaft für Mathematik und Datenverarbeitung Nr. 111 München, Wien 1979, S. 63-76
Petri, Carl Adam, Zur "Vermenschlichung" des Computers, in: GMD-Spiegel Nr.3/4,1983, S.42-44
Pflüger, Jörg, Informatik vor dem Gesetz, in: Coy, Wolfgang u.a. (Hrsg.), Sichtweisen der Informatik, Braunschweig, Wiesbaden 1992, S. 277-298
Pflüger, Jörg, Über die Verschiedenheit des maschinellen Sprachbaues, in: Bolz, Norbert, Kittler, Friedrich A.; Tholen, Christoph (Hrsg.), Computer als Medium, München 1994, S. 161-181
Pflüger, Jörg, Informatik auf der Mauer, in: Informatik Spektrum 17 (1994) 6, S. 251-257
Pflüger, Jörg, Leitbilder der Programmiersprachenentwicklung, in: Friedrich, Jürgen; Herrmann, Thomas; Peschek, Max; Rolf, Arno (Hrsg.), Informatik und Gesellschaft, Heidelberg, Berlin, Oxford 1995, S. 196-210
Pflüger, Jörg, Writing, Building, Growing: Leitvorstellungen der Programmiergeschichte, in: Hellige, Hans Dieter (Hrsg.), Geschichten der Informatik. Visionen, Paradigmen und Leitmotive, Berlin, Heidelberg, New York 2004, S. 275-320
Pflüger, Jörg, Konversation, Manipulation, Delegation. Zur Ideengeschichte der Interaktivität, , in: Hellige, Hans Dieter (Hrsg.), Geschichten der Informatik. Visionen, Paradigmen und Leitmotive, Berlin, Heidelberg, New York 2004, S. 367-408
Pilorge, R., Comprendre l'informatique, Paris 1969
Piloty, Robert, Betrachtungen über das Problem der Datenverarbeitung, in: Hoffmann, Walter; Walther, Alwin (Hrsg.), Elektronische Rechenmaschinen und Informationsverarbeitung (Nachrichtentechnische Fachberichte, Bd. 4), Braunschweig 1956, S. 5-8
Pollack, Seymour Victor, The Development of Computer Science, in: ders. (Hg.), Studies in Computer Science (Studies in Mathematics, Bd. 22, 1982),, S. 1-51 und Introduction, S. VII-XV
Ponte, Maurice; Braillard, Pierre, L'informatique, Paris 1970
Prasse, Michael; Rittgen, Peter, Bemerkungen zu Peter Wegners Ausführungen über Interaktion und Berechenbarkeit, in: Informatik-Spektrum 21 (1998), S. 141-146
Ralston, Anthony, Introduction to Programming and Computer Science, New York u.a. 1971; Neudruck Huntington, N. Y. 1978 (A kyb 400e/194)
Ralston, Anthony, What is Computer Science, in: ders., Introduction to Programming and Computer Science, New York u.a. 1971; Neudruck Huntington, N. Y. 1978, S. 1-7
Raymond, Francois Henri, L'automatique des informations, Paris 1957
Rechenberg, Peter, Was ist Informatik?, München, Wien 1991; 3. Aufl. 2000
Reuter, Andreas, Was lehren wir eigentlich, wenn wir Informatik lehren? in: Desel, Jörg (Hrsg.), Das ist Informatik, Berlin, Heidelberg, New York 2001, S. 33-46
Rice, J. K.; Rice, J. R., Introduction to Computer Science, New York 1969
Rolf, Arno, Informatik, neue „Informations- und Kommunikationstechniken und sozialorientierte Technikforschung in der „Informationsgesellschaft", in: ders. (Hrsg.), Neue Techniken Alternativ. Möglichkeiten und Grenzen sozialverträglicher Informationstechnikgestaltung, Hamburg 1986, S. 99-125
Rolf, Arno, Sichtwechsel. Informatik als Gestaltungswissenschaft, in: Coy, Wolfgang u.a. (Hrsg.), Sichtweisen der Informatik, Braunschweig, Wiesbaden 1992, S. 33-47
Rolf, Arno, Das Selbstverständnis der Informatik, in: Friedrich, Jürgen; Herrmann, Thomas; Peschek, Max; Rolf, Arno (Hrsg.), Informatik und Gesellschaft, Heidelberg, Berlin, Oxford 1995, S. 3-7
Rolf, Arno, Neue Sichtweisen der Informatik, in: Friedrich, Jürgen; Herrmann, Thomas; Peschek, Max; Rolf, Arno (Hrsg.), Informatik und Gesellschaft, Heidelberg, Berlin, Oxford 1995, S. 8-14

Roos, Paul, Theoretisches und gesellschaftliches Interesse der Informatik, in: Dirlewanger, Werner u.a., Einführung in Teilgebiete der Informatik I, Berlin, New York 1972, S. 9-24
Rozenberg, Grzegorz, Carl Adam Petri und die Informatik, in GMD-Spiegel (1991) 3/4, S. 52-57
Ruyer, Raymond, La Cybernétique et l'origine de l'information, Paris 1954
Salton, Gerard, Information Science in a Ph.D. Computer Science Program.in: Commun. ACM 12 (1969) 2, S. 111-117
Salton, Gerard, What is Computer Science? in: Journal of the ACM 19 (1972) 1, S.
Salton, Gerard, Some Characteristics of Future Information Systems, in: SIGIR Forum 18 (1985) 2-4, S. 28-39
Samelson, Klaus, Entwicklungslinien der Informatik, in: GI-8. Jahrestagung 1978 (Informatik-Fachberichte, Bd. 16), Berlin, Heidelberg, New York 1978, S. 132-148
Scharf, Joachim-Hermann, Informatik. Vorträge anläßlich der Jahresversammlung vom 14. bis 17. Oktober 1971 zu Halle (Saale), in: Nova Acta Leopoldina, Neue Folge Nr. 206, Bd. 37/1 (1972), Leipzig 1972
Schelhowe, Heidi, Das Medium aus der Maschine. Zur Metamorphose des Computers. Frankfurt a. M., New York 1997
Schelhowe, Heidi, Produktionsmaschine oder Kommunikationsmedium? Carl Adam Petri und sein Programm einer einheitlichen Theorie der Computertechnologie, erscheint in: Hellige, Hans Dieter (Hrsg.), Geschichten der Informatik. Visionen, Paradigmen und Leitmotive, Berlin, Heidelberg, New York 2003
Schinzel, Britta (Hrsg.), Schnittstellen – Zum Verhältnis von Informatik und Gesellschaft, Braunschweig, Wiesbaden 1996
Schinzel, Britta (Hrsg.), Interdisziplinäre Informatik: Neue Möglichkeiten und Probleme für die Darstellung und Integration komplexer Strukturen in verschiedenen Feldern der Neurologie. Freiburger Universitätsblätter, Freiburg 2001
Schinzel, Britta; Kleinn, Karin, Quo vadis, Informatik ?, in: Informatik Spektrum, 24 (2001), S. 91-97
Seetzen, Jürgen, Information, Kommunikation, Organisation. Anmerkungen zur „Theorie der Informatik, in: Coy, Wolfgang u.a. (Hrsg.), Sichtweisen der Informatik, Braunschweig, Wiesbaden 1992, S. 83-96
Siefkes, Dirk, Computer Science as Cultural Development. Toward a Broader Theory, in: Freksa, Christian; Jantzen, Matthias; Valk, Rüdiger (Hrsg.), Foundations of Computer Science. Potential, Theory, Cognition, (Lecture Notes in Computer Science, Bd. 1337), Berlin, Heidelberg, New York 1997, S. 37-47
Siefkes, Dirk; Eulenhöfer, Peter; Stach, Heike; Städtler, Klaus (Hrsg.), Sozialgeschichte der Informatik. Kulturelle Praktiken und Orientierungen, Wiesbaden 1998
Spencer, Donald D., Fundamentals of Computers, Indianapolis 1969
Steinbuch, K. (Hg.) (1962), Taschenbuch der Nachrichtenverarbeitung, Berlin, Heidelberg, New York; erst die 3. Aufl. von 1974 in 3 Bänden unter dem Titel: Taschenbuch der Informatik
Steinbuch, Karl; Schmid, Detlef, Zur Zukunft der Informatik, in: Ernst Schmacke (Hrsg.), 1980 ist morgen, Düsseldorf 1969, S. 52-74
Steinmüller, Wilhelm, Informationstechnologie und Gesellschaft - Eine Einführung in die Angewandte Informatik, Darmstadt 1993
Syrbe, Max, Über die Notwendigkeit einer Systemtheorie in der Wissenschaftsdisziplin Informatik, in: Informatik-Spektrum 18 (1995), S. 222-227
Thompson, Frederick B., The Need for a Science of Information, in: Rosenthal, Paul H.; Mish, Russell K., Multi-Access Computing. Modern Research and Requirements, Rochelle Park, N. J. 1974, S. 211-218
Valk, Rüdiger, Zum Selbstverständnis der Informatik, in: Informatik Spektrum 20 (1997) 2, S.
Valk, Rüdiger, Die Informatik zwischen Formal- und Humanwissenschaften in: Informatik Spektrum 20 (1997) 2, S. 95-100
Vollmar, Roland; Ritter, Helge; Claus, Volker (Hrsg.), Berechnungsmodelle, in: Wilhelm, Reinhard (Hrsg.), Informatik – Grundlagen, Anwendungen, Perspektiven, München 1996, S. 105-113
Walker, T. M.; Cotterman, W. W., An Introduction to Computer Science and Algorythmic Processes, Boston 1970
Wedekind, H.; Görz, G.; Kötter, R.; Inhetveen, R., Modellierung, Simulation, Visualisierung. Zu aktuellen Aufgaben der Informatik, in: Informatik Spektrum, 21 (1998), S. 265-272
Wegner, Peter, Three Computer Cultures – Computer technology, Computer mathematics, and Computer Science, in: Computing Surveys 10 (1970), S.

Wegner, Peter, Why Interaction Is More Powerful Than Algorithms, in: Communications of the ACM 40 (1997) 5, S.81-91

Wegner, Peter, Dennis, Jack; Hammer, Michael; Teichroew, D. (Hrsg.), Research Directions in Software Technology, Cambridge, Mass., London 1979

Wilhelm, Reinhard (Hrsg.), Informatik – Grundlagen, Anwendungen, Perspektiven, München 1996

Wilhelm, Reinhard (Hrsg.), Informatics. 10 Years Back(Lecture Notes in Computer Science, Bd. 2000), Berlin, Heidelberg, New York 2001

Winograd, Terry; Flores, Fernando, Understanding Computers and Cognition: a New Foundation for Design, Norwood, N. J. 1986

Zadeh, Lofti A., The Dilemma of Computer Sciences, in: A. Finerman (Hg.), University Education in Computing Science, New York, London 1968, S. 61-68

Zadeh, Lofti A., Applied Computer Science, in: A Panel Session – Education of Computer Professional, in: AFIPS, Bd. 34, Spring Joint Computer Conference 1969, S. 539-540

Zemanek, Heinz, Was ist Informatik?, in: Elektronische Rechenanlagen 13 (1971) 4, S. 157-161

Zemanek, Heinz, Informationsverarbeitung und die Geisteswissenschaften, Wien 1987

Zemanek, Heinz, Informationsverarbeitung und Geisteswissenschaften. Vortrag in der Gesamtsitzung der Österreichischen Akademie der Wissenschaften am 3. $. 1987, Anzeiger der phil.-hist. Klasse der ÖAW 124 (1988), S. 199-225; wiedergedruckt in: ders., Ausgewählte Beiträge zu Geschichte und Philosophie der Informationsverarbeitung (Schriftenreihe der Österreichischen Computer Gesellschaft, Bd. 43), Wien, München 1988, S 129-156

Zemanek, Heinz, Die Entwicklung der logischen Basis der Computerwissenschaften, in: 9. Technikgeschichtliche Tagung der Stiftung Eisenbibliothek, FERRUM 58 (1987), S. 34-46; wiedergedruckt in: ders., Ausgewählte Beiträge zu Geschichte und Philosophie der Informationsverarbeitung (Schriftenreihe der Österreichischen Computer Gesellschaft, Bd. 43), Wien, München 1988, S. 37-53

Zemanek, Heinz, Ausgewählte Beiträge zu Geschichte und Philosophie der Informationsverarbeitung (Schriftenreihe der Österreichischen Computer Gesellschaft, Bd. 43), Wien, München 1988 (A kyb 007/970)

Zemanek, Heinz, Grenzen des Computers, in: Franz, Herbert; Fritsch, Gerolf; Kozdon, Baldur, An den Grenzen der Machbarkeit, S. 123-137; wiedergedruckt, in: Zemanek, Heinz, Ausgewählte Beiträge zu Geschichte und Philosophie der Informationsverarbeitung (Schriftenreihe der Österreichischen Computer Gesellschaft, Bd. 43), Wien, München 1988, S. 99-127

Zemanek, Heinz, Weltmacht Computer. Weltreich der Information, Esslingen, München 1991

Zemanek, Heinz, Hat die Informatik den Computer im Griff? Mobilisierung zur Stabilisierung, in: Langenheder, Werner; Müller, Günter; Schinzel, Britta, Informatik cui bono? GI-FB 8 Tagung Freiburg, 23.- 26. September 1992, Berlin, Heidelberg, New York 1992, S. 20-28

Zemanek, Heinz, Hardware - Software. An Equivalence and a Contradiction, in: Freksa, Christian; Jantzen, Mattias; Valk, Rüdiger (Hrsg.), Foundations of Computer Science. Potential, Theory, Cognition, (Lecture Notes in Computer Science, Bd. 1337), Berlin, Heidelberg, New York 1997, S. 9-19

Zuse, Konrad, Grundsätzliche Gedanken für die Entwicklung leistungsfähiger programmgesteuerter Rechengeräte auf weite Sicht, Ms., Zuse-Papiere Paderborn 039/003; Zuse-Internet-Archiv: http://www.zib.de/zuse/Inhalt/Texte/Chrono/60er/Html/0304/0304.html

Bibliographie zur Geschichte der Informatikdisziplin Computerarchitektur

Abrams, Marshall D.; Stein, Philip G., Computer Hardware and Software. An Interdisciplinary Introduction, Reading, Mass, Menlo Park, Cal., London 1973

Aiken, Howard H., Proposed Automatic Calculating Machine, masch.schriftl. Fassung November 1937); mit leichten veränderungen in: IEEE Spectrum 1 (1964) 8, S. 62-69; wiedergedruckt in: Pylyshyn, Zenon W., (Hrsg.), Perspectives On the Computer Revolution, Englewood Cliffs, N.J. 1970, S. 29-36; in: Randell, Brian (Hrsg.), The Origins of Digital Computers. Selected Papers, Berlin, Heidelberg, New York 1973, S. 191-197; 2. Aufl. 1975; 3. Aufl. 1982, S. 195-201; Neudruck nach der Originalfassung in: Cohen, I. Bernard; Welch, Gregory M.; Campbell, Robert V. D. (Hrsg.), "Making Numbers": Howard Aiken and the Computer, Cambridge, MA, London 1999, S.

Aiken, Howard H., The Automatic Sequence Controlled Calculator, Lecture 13, in: Campbell-Kelly, Martin; Williams, Michael R. (Hrsg.), The Moore School Lectures. Theory and Techniques for Design of Electronic Digital Computers (Charles Babbage Institute (Hrsg), Reprint Series for the History of Computing, Bd. 9), London, Los Angeles, San Francisco 1985, S. 149-168

Aiken, Howard H.; Hopper, Grace M., The Automatic Sequence Controlled Calculator, in: Electrical Engineering 65 (1946), Teil I: (Sept.), S.384-391; Teil II: (Okt.), S. 449-454; Teil III: (Nov.), 522-528; wiedergedruckt in: Randell, Brian (Hrsg.), The Origins of Digital Computers. Selected Papers, Berlin, Heidelberg, New York 1973, S. 199-218; 2. Aufl. 1975; 3. Aufl. 1982, S. 203-222

Alexander, Christopher, Notes on the Synthesis of Form, Cambridge, Mass. 1964

Amdahl, Gene M., Logical Design Method, in: AFIPS 13 (1958) WJCC, S. 167-

Amdahl, Gene M., New Concepts in ComputingSystem Design, in: Proceedings of the IRE 50 (1962) 5, S. 1073-1077

Amdahl, Gene M., Microprogramming and Stored Logic, in: Datamation (1964) 2, S. 24-27

Amdahl, Gene M., Architectural Questions of the Seventies, in: Datamation 16 (1970) 1, S. 66-68

Amdahl, Gene M., Archtitectural Concepts for High-Performance, General-Purpose Computers, in: Information Processing 1983, Proceedings of IFIP Congress '83, Amsterdam u. a. 1983, S. 369-373

Amdahl, Gene M; Blaauw, Gerrit A.; Brooks Jr., Frederick P., Architecture of the IBM System /360, in: IBM Journal of Research and Development 8 (1964) April, S. 86-101

Amdahl, Gene M.; Padegs, A., The Structure of the System /360. Part I: Outline of Logical Structure, in: IBM Systems Journal 3 (1964) 2, S. 117-196

Anagnostopoulous, P. C.; Michel, M .J.; Sockut, G. H.; Stabler, G. M.; Van Dam, Andries, Computer Architecture and Instruction Set Design, in: AFIPS X (1972), S. 519-527

Anderson, D. W.; Sparacio, F. J.; Tomasulo, R. M., The IBM System/360 Model 91: Machine Philosophy and Instruction Handling, in: IBM Journal of Research and Development 11 (1967) 1, S. 8-24

Anderson, George A.; Jensen, E. Douglas, Computer Interconnection: Taxonomy, Characteristics, and Examples, in: ACM Computing Surveys, Dez. 1975

Andrews, Ernest G., Bell Laboratories Digital Computers, in: Bell Laboratories Record, 35 (1957), 81-84

Andrews, Ernest G., Telephone Switching and the Early Bell Laboratories Computers, in:

Andrews, Ernest G.; Arsenault, William R.; Huskey, Harry D., General-Purpose Computers, in: Huskey, Harry D.; Korn, Granino A. (Hrsg.), Computer Handbook, New York, San Francisco, Toronto, London 1962, 20-1-34

Apfelbaum, Henry; Case, Richard P.; Julissen, J. Egil; Shriver, Bruce; Stone, Harold S.; Thornton, James E., Computer System Organization: Problems of the 1980's, in: IEEE Computer (1978) 9, S. 20-28

Aspray, William und Arthur W. Burks (Hrsg.), Papers of John von Neumann on Computing and Computing Theory, Cambridge/Mass., London, Los Angeles, San Francisco 1987

Aspray, William, From Mathematical Constructivity to Computer Science: Alan Turing, John von Neumann, and the Origins of Computer Science and Mathematical Logic, Phil. Diss. University of Wisconsin, Madison 1980

Aspray, William, The Stored Program Concept, in: IEEE Spectrum 27 (1990) 9, S. 51-

Aspray, William, John von Neumann, and the Origins of Modern Computing, Cambridge, MA, London 1990

Astrahan, Morton M.; Rochester, The Logical Organization of the New IBM Scientific Calculator, in: Proceedings of the ACM, Mai 1952, S.

Babbage, Charles, On the Economy of Machinery and Manufacture (1835), 4. Aufl., in: Campbell-Kelley, Martin (Hrsg.), Works of Babbage, Bd. 5, New York 1989

Baer, Jean-Loup, Computer Systems Architecture, Potomac, MD 1980

Baer, Jean-Loup, Computer Architecture, in: IEEE Computer (1984) 10, S. 77-87

Ballance, Richard S.; Cocke, John; Kolsky, Harwood G., The Look-Ahead Unit, in: Buchholz, Werner (Hrsg.), Planning a Computer System. Project Stretch, New York, Toronto, London 1962, S. 228-247

Banks, A. W., The Logical Design of an Idealized General Purpose Computer, in: Journal of the Franklin Institute (Mrz. 1956), S. 421-436

Banse, Gerhard; Wendt, Helge, Erkenntnismethoden in den Technikwissenschaften, Berlin 1986

Barnum, A.; Knapp, M., Proceedings of 1962 Workshop on Computer Organization, Washington, D. C. 1963

Baron, Robert J.; Higbie, Lee, Computer architecture, Reading, Mass. u.a. 1992

Bartee, Thomas C., Digital Computer Fundamentals, New York 1960; 2. Aufl. 1966; 3. Aufl. Tokyo, Düsseldorf 1972 (A 19 H kyb 325 e 998-3a)

Bartee, Thomas C., Computer Architecture and Logic Design, New York 1991

Bartee, Thomas C.; Lebow, Irwin L.; Reed, Irving S., Theory and Design of Digital Machines, New York, San Francisco, Toronto, London 1962

Barton, R. S., A New Approach to the Functional Design of a Digital Computer, in: AFIPS 19 (1961) WJCC, S.393-396

Barton, R. S., Ideas for Computer Systems Organization - a Personal Survey, in: Tou, J. T.(Hrsg.), Software Engineering. Proceedings of the Third Symposium on Computer and Information Sciences Held in Miami Beach, Florida, Dezember 1969, New York u.a. 1970, Vol 1, S. 7-16

Bashe, Charles J.; Johnson, Lyle R.; Palmer, John H.; Pugh, Emerson W., The Architecture of IBM's Early Computers, in: IBM Journal of Research and Development 25 (1981) 5, S. 363-375

Bauer, Walter F., Computer Design From the Programmer's Viewpoint, in: AFIPS 14 (1958) EJCC, S. 46-51

Bauer, Walter F., Modern Large Scale Computer System Design, in: Computers and Automation 6 (1957) 1, S. 8-13

Bauer, Walter F.; Granholm, D. L. J. W., Advanced Computer Applications, in: Proceedings of the IRE, Jan.1961, S.301-304

Beckman, F. S.; Brooks Jr., Frederick P.; Lawless, W. J. Jr., Developments in the Logical Organization of Computer Arithmetic and Control Units, in: Proceedings of the IRE 49 (1961) 1, S. 53-66

Beizer, Boris, The Architecture and Engineering of Digital Computer Complexes, 2 Bde. New York, London 1971

Bell, C. Gordon, Computer Architecture: Comments on the State-of-the Art, in: 3. Jahrestagung der Gesellschaft für Informatik, Hamburg 1973, (Lecture Notes in Computer Science, Bd. 1), Berlin, Heidelberg 1973, S. 18-24

Bell, C. Gordon, What Have We Learned from the PDP-11? in: Boulaye, Guy G.; Lewin, Douglas W. (Hrsg.), Computer Architecture. Proceedings of the NATO Advanced Study Institute, St. Raphael, Sept. 1976 (NATO Advanced Study Institutes Series, Serie C, Bd. 32), Dordrecht, Boston 1977, S. 1-38

Bell, C. Gordon, What Have We Learned from the PDP-11? in: Jones, Anita K., Perspectives on Computer Science. From the 10th Anniversary Symposium at the Computer Science Department, Carnegie-Mellon University, New York, San Francisco, London 1977, S. 7-38

Bell, C. Gordon, Computer Structures: What Have We Learned from the PDP-11? in: Proceedings Third Annual Symposium on Computer Architecture, IEEE, Pittsburgh, Penn. Januar 1976, New York 1976, in: Computer Architecture News, Bd. 4, Nr. 4, S. 1-13

Bell, C. Gordon; Eggert, J. L.; Grason, J.; Williams, P., The Description and Use of Register Transfer Modules (RTM's), in: IEEE Transactions on Computers, C-21, 21 (1972) 5, S. 495-500

Bell, C. Gordon; Grason, J., Register transfer modules (RTM) and their Design, in: Computer Design (Mai 1971). S. 87-94 .

Bell, C. Gordon; Grason, J.; Newell, Allen, Designing Computers and Digital Systems, Using PDP16 Register Transfer Modules, Maynard, Mass. 1972

Bell, C. Gordon; Mudge, J. Craig; McNamara, John E., Computer Engineering: A DEC View of Hardware System Design, Bedford, MA 1978

Bell, C. Gordon; Newell, Allen, The PMS and ISP Descriptive Systems for Computer Structures, in: AFIPS, Bd. 36 (SJCC 1970), S. 351-374

Bell, C. Gordon; Newell, Allen, Computer Structures: Readings and Examples, New York, St Louis, San Francisco 1971; (2. Aufl. New York, St. Louis, San Francisco 1982, (A kyb 326e/426) (2. Aufl. siehe Sieworek, Daniel P. u. a.)

Bell, C. Gordon; Newell, Allen, Computer Structures: Past, Present, and Future, in: AFIPS, Bd. 39 (FJCC 1971), S. 387-394

Bell, C. Gordon; Strecker, William D., Computer Structures: What Have We Learned from the PDP-11? in: Proceedings Third Annual Symposium on Computer Architecture, IEEE, Pittsburgh, Penn. Januar 1976, New York 1976, in: Computer Architecture News, Bd. 4, Nr. 4, S. 1-14

Bennett, J. M.; Glennie, Alec E., Programming for High-speed Digital Calculating Machines, in: Bowden, Bertram Vivian, (Hrsg.), Faster Than Thought. A Symposium on Digital Computing Machines, New York, London 1953, S. 101-113

Bennett, J. M., Digital Computers and the Engineer, in: Bowden, Bertram Vivian, (Hrsg.), Faster Than Thought. A Symposium on Digital Computing Machines, New York, London 1953, S. 223-233

Bensky, L. S., Block Diagrams in Logic Design, in: AFIPS, Bd. 13 (WJCC 1958), S. 177-178

Bergman, G. D., Electronic Architectures for Digital Processing: Software/Hardware Balance in Real-time Systems, New York u.a. 1992

Berkeley, Edmund Callis; Wainwright, Lawrence, Computers. Their Operation and Applications, New York, Amsterdam, London 1956; 11 Drucke bis 1968

Bernus, P.; Mertins, K.; Schmidt, G., (Hrsg.), Handbook of Architectures of Information Systems, Berlin, Heidelberg, New York 1998

Biermann, Alan W., Great Ideas in Computer Science: A Gentle Introduction, Cambridge u.a. 1990

Blaauw, Gerrit A., The Structure of the System /360. Part V: Multisystem Organization, in: IBM Systems Journal 3 (1964) 2, S. 181-195

Blaauw, Gerrit A., Multisystem-Organization. Aufbau des Systems /360, in: IBM System Journal 3 (1964) 2, S. 79-97

Blaauw, Gerrit A., Hardware Requirements for the Fourth Generation Computers, in: Gruenberger, Fred J.(Hrsg.), Fourth Generation Computers: User Requirements and Transition, Englewood Cliffs, N. J. 1970, S. 155-168

Blaauw, Gerrit A., Computer Architecture, in: Elektronische Rechenanlagen 14 (1972) 4, S. 154-159; wiedergedruckt in: Hasselmeier, H.; Spruth, W. G., Rechnerstrukturen. Vorträge des Informatik-Symposiums der IBM Deutschland, Wildbad 1973, München, Wien 1974, S. 14-35

Blaauw, Gerrit A., Digital System Imlementation, Englewood Cliffs 1976

Blaauw, Gerrit A.; Brooks Jr., Frederick P., The Structure of the System /360. Part I: Outline of Logical Structure, in: IBM Systems Journal 3 (1964) 2, S. 119-135; wiedergedruckt in: Bell, C. Gordon; Newell, Allen, Computer Structures: Readings and Examples, New York, St Louis, San Francisco 1971, S. 588-601

Blaauw, Gerrit A.; Brooks Jr., Frederick P., Computer Architecture. Concepts and Evolution, Reading, MA, Harlow, Menlo Park, CA 1997

Blaauw, Gerrit A.; Händler, Wolfgang (Hrsg.), Workshop on Taxonomy in Computer Architecture, Universität Erlangen-Nürnberg 1981

Bloch, Erich The Engineering Design of the Stretch Computer, in: AFIPS 16 (1959) EJCC, S.48-59, wiedergedruckt in: Bell, C. Gordon; Newell, Allen (Hrsg.), Computer Structures: Readings and Examples, New York, St. Louis, San Francisco 1971, S. 421-439; überarbeitet in:

Bloch, Erich The Central Processing Unit, in: Buchholz, Werner (Hrsg.), Planning a Computer System. Project Stretch, New York, Toronto, London 1962, S. 202-227 (überarbeitete Version von: The Engineering Design of the Stretch Computer)

Bode, Arndt (Hrsg.), RISC-Architekturen, 2. Aufl. Mannheim, Wien Zürich 1990

Bode, Arndt (Hrsg.), Parallel Computer Architectures: Theory, Hardware, Software, Applications (Lecture Notes in Computer Science ; 732), Berlin, Heidelberg, New York 1993

Bode, Arndt; Händler, Wolfgang, Rechnerarchitektur. Grundlagen und Verfahren, Heidelberg, New York 1980

Bolch, Gunter, Leistungsbewertung von Rechensystemen mittels analytischer Warteschlangenmodelle, Stuttgart 1989

Booth, Andrew D.; Booth, Kathleen H. V., Automatic Digital Calculators, London 1953; 2. Aufl. London 1956; 3. Aufl. 1965

Boulaye, Guy G., Structured Design for Structured Computer Architecture, in: Händler, Wolfgang (Hrsg.) Computer Architecture. Workshop of the Gesellschaft für Informatik, Erlangen, Mai 1975 (Informatik-Fachberichte 4), Berlin, Heidelberg, New York 1976, S. 9-22

Boulaye, Guy G., Structured Design, in: ders.; Lewin, Douglas W. (Hrsg.), Computer Architecture. Proceedings of the NATO Advanced Study Institute, St. Raphael, Sept. 1976 (NATO Advanced Study Institutes Series, Serie C, Bd. 32), Dordrecht, Boston 1977, S. 39-48

Boulaye, Guy G.; Lewin, Douglas W. (Hrsg.), Computer Architecture. Proceedings of the NATO Advanced Study Institute, St. Raphael, Sept. 1976 (NATO Advanced Study Institutes Series, Serie C, Bd. 32), Dordrecht, Boston 1977

Bowden, Bertram Vivian, The Organization of a Typical Machine, in: ders., (Hrsg.), Faster Than Thought. A Symposium on Digital Computing Machines, New York, London 1953, S. 67-77

Bowden, Bertram Vivian, The Construction, Performance and Maintanence of Digital Computers, in: ders., (Hrsg.), Faster Than Thought. A Symposium on Digital Computing Machines, New York, London 1953, S. 78-100

Bowden, Bertram Vivian, The Application of Digital Computers to Business and Commerce, in: ders., (Hrsg.), Faster Than Thought. A Symposium on Digital Computing Machines, New York, London 1953, S. 246-271

Bowden, Bertram Vivian, Thought and Machine Processes, in: ders., (Hrsg.), Faster Than Thought. A Symposium on Digital Computing Machines, New York, London 1953, S. 311-337

Brainerd, John Grist; Sharpless, T. Kite, The ENIAC, in: Electrical Engineering 67 (1948) Febr., S. 163-172; wiedergedruckt in: in: Proceedings of the IEEE 87 (1999) 6, S. 1031-1041

Breuer, M. A. (Hrsg.), Design Automation of Digital Systems (Theory and Techniques, Bd. 1), Englewood Cliffs, N. J. 1972

Brooks Jr., Frederick P.,The Analytic Design of Automatic Data Processing Systems, Phil. Diss. Harvard University 1956

Brooks Jr., Frederick P., A Program Controlled Program Interruption System, in: AFIPS 12, Proceedings of the IRE/AIEE/ACM EJCC, (1957), S.128-132; verarbeitet in: Buchholz, Werner (Hrsg.), Planning a Computer System. Project Stretch, New York, Toronto, London 1962, S. 133-149

Brooks Jr., Frederick P., Architectural Philosophy, in: Buchholz, Werner (Hrsg.), Planning a Computer System. Project Stretch, New York, Toronto, London 1962, S. 5-16

Brooks Jr., Frederick P., The Execute Operations: A Fourth Mode of Instruction Sequencing, in: Communications of the ACM 3 (1960) 3, S. 168-170; verarbeitet in: Buchholz, Werner (Hrsg.), Planning a Computer System. Project Stretch, New York, Toronto, London 1962, S. 133-149

Brooks Jr., Frederick P., Management Problems of the Business Computer Installation, in: IRE Transactions on Engineering Management, Bd. EM-3, Nr, 3, S. 60-63

Brooks Jr., Frederick P., Recent Developments in Computer Organization, in: Advances in Electronics 18 (1963), S. 45-65

Brooks Jr., Frederick P., The Future of Computer Architecture, in: Information Processing 1965, Proceedings of IFIP Congress '65, 2 Bde. Washington, D. C., London 1965, Bd. 2, S. 87-91

Brooks Jr., Frederick P., The Mythical Man-Month. Essays in Software Engineering, Reading, MA, Menlo Park, CA., London 1975

Brooks Jr., Frederick P., No Silver Bullet. Essence and Accidents of Software Engineering, in IEEE Computer 20 (1987), S. 10-19

Brooks Jr., Frederick P., The Mythical Man-Month. Essays in Software Engineering, 2. Aufl Reading, Ma, Menlo Park, Cal., London 1995

Brooks Jr., Frederick P.; Blaauw, Gerrit A.; Buchholz, Werner, Processing Data in Bits and Pieces, in: IRE Transactions on Electronic Computers, EC-8, 8 (1959) 2, S. 118-124; und in: Information Processing, Proceedings of the International Confernence on Information Processing, Paris Juni 1959, München, London 1960, S. 375-382

Brooks Jr., Frederick P.; Iverson, Kenneth E., Automatic Data Processing, New York, London, Sydney 1963

Brown, E. L., Digital Computer Design, New York 1963

Buchheim, Gisela, Zur Herausbildung der Technikwissenschaften, in: Dresdner Beiträge zur Geschichte der Technikwissenschaften, Heft 1 (1980), TU Dresden

Buchheim, Gisela; Sonnemann, Rolf, Geschichte der Technikwissenschaften, Leipzig 1990
Buchholz, Werner, Selection of an Instruction Language, in: AFIPS Bd. 13, Proceedings of the IRE/AIEE/ACM WJCC, 1958, S. 128-130; überarbeitet in: ders. (Hrsg.), Planning a Computer System. Project Stretch, New York, Toronto, London 1962, S. 122-132
Buchholz, Werner, Fingers or Fists? (The Choice of decimal or binary Numbers, in: Communications of the ACM 2 (1959) 2, S. 3-11
Buchholz, Werner (Hrsg.), Planning a Computer System. Project Stretch, New York, Toronto, London 1962 (A Kyb 326/295)
Buchholz, Werner, System Summary of IBM 7030, in: ders. (Hrsg.), Planning a Computer System. Project Stretch, New York, Toronto, London 1962, S. 17-32
Buchholz, Werner, Choosing a Number Base, in: ders. (Hrsg.), Planning a Computer System. Project Stretch, New York, Toronto, London 1962, S. 42-59 (Neufassung von: Fingers or Fists?)
Buchholz, Werner, Input-Output Control, in: ders. (Hrsg.), Planning a Computer System. Project Stretch, New York, Toronto, London 1962, S. 179-191
Buchholz, Werner; Gunning, William F.; Huskey, Harry D.; Thorensen, Ragnar, Digital-Computer-System Design, in: Huskey, Harry D.; Korn, Granino A. (Hrsg.), Computer Handbook, New York, San Francisco, Toronto, London 1962, 16-1-34
Bulman, D. M., Stack Computers, in: IEEE Computer 10 (1977) 5, S. 14-16
Bundschuh, Bernd; Sokolowsky, Peter, Rechnerstrukturen und Rechnerarchitekturen : Grundlagen - sequentielle Systeme - innovative Architekturen, 1. Auf. 1988; 2., erw. und überarb. Aufl. Braunschweig, Wiesbaden 1996 (A inf 200 eb/511)
Burd, Stephen D., Systems Architecture; Hardware and Software in Information Systems 1993
Burks, Arthur W.; Copi, I. M., The Logical Design of an Idealized General Purpose Computer, in: Journal of the Franklin Institute 261 (1956) , Teil I=S. 249-; Teil II=S. 421-
Burks, Arthur W., From ENIAC to the Stored-Program Computer: Two Revolutions in Computers, in: Metropolis, Nicholas C.; Howlett, Jack; Rota, Gian-Carlo (Hrsg.), A History of Computing in the Twentieth Century. A Collection of Essays, New York, London 1980, S. 311-344
Burks, Arthur W.; Burks, Alice R., The ENIAC: First General-Purpose Electronic Computer, in: Annals of the History of Computing, 3 (1981) 4, S. 310-399
Burks, Arthur W., Goldstine, Herman H.; Neumann, John von, Prelimininary Discussion of the Logical Design of an Electronic Computing Instrument, Part I, Volume 1, Institute for Advanced Study Princeton, N. J. Juni 1946; 2. Aufl. 1947; wiedergedruckt in: Datamation 8 (1962) 9, S. 24-31 (Teil I); Datamation 8 (1962) 10, S. 36-41 (Teil I); in: Taub, Abraham H. (Hrsg.), Complete Works of John von Neumann, 6 Bde.Oxford (UK), New York 1961-63, Bd. 5, S. 34-79; in: Randell, Brian (Hrsg.), The Origins of Digital Computers. Selected Papers, Berlin, Heidelberg, New York 1973, S. 371-385; 2. Aufl. 1975; 3. Aufl. 1982, S. 399-413 (gekürzt); ebenso in: Bell, C. Gordon; Newell, Allen, Computer Structures: Readings and Examples, New York 1971, S. 92-119; in: Aspray, William; Burks, Arthur W. (Hrsg.), Papers of John von Neumann on Computing and Computing Theory, Cambridge/ Mass.-London-Los Angeles-San Francisco 1987, S. 97 - 142
Caldwell, Samuel H., Switching Circuits and Logical Design, New York 1958
Campbell, Sullivan G., Floating Point Operation, in: Buchholz, Werner (Hrsg.), Planning a Computer System. Project Stretch, New York, Toronto, London 1962, S. 92-121
Campbell-Kelly, Martin; Aspray, William, Computer. A History of the Information Machine, New York 1996
Campbell-Kelly, Martin; Williams, Michael R. (Hrsg.), The Moore School Lectures. Theory and Techniques for Design of Electronic Digital Computers. Lectures given at Moore School of Electrical Engineering, University of Pennsylvania Philadelphia, July 8-August 31, 1946, Reprint in: Charles Babbage Institute (Hrsg), Reprint Series for the History of Computing, Bd. 9, London, Los Angeles, San Francisco 1985
Carlson, C. B., A Survey of High-Level Language Computer architecture, in: Chu, Yaohan (Hrsg.), High-Level Language Computer Architecture, New York , London. 1975, S. 31-62
Carpinelli, John D., Computer Systems Organization & Architecture, Addison-Wesley 2000
Case, R. P.; Padegs, A., Architecture of the IBM System /370, in: Communications of the ACM, 21 (1978) 1, S. 73-96; wiedergedruckt in: Sieworek, Daniel P.; Bell, C. Gordon; Newell, Allen, Computer Structures: Principles and Examples, New York 1982, S. 830-855

Ceruzzi, Paul E., Crossing the Divide: Architectural Issues and the Emergence of the Stored Program Computer, 1935-1955, in: Annals of the History of Computing, 19 (1997) 1, S. 5-12 ; Netzversion: http://web.mit.edu/STS.035/www/PDFs/ceruzzi.pdf
Ceruzzi, Paul E., A History of Modern Computing, Cambridge, MA 1998
Ceruzzi, Paul E., "Nothing New Since von Neumann" A Historian Looks at Computer Architecture, 1945-1995, in: Rojas, Raúl; Hashagen, Ulf (Hrsg.), The First Computers - History and Architectures, Cambridge, MA, London 2000, S. 195-217
Chalk, B. S., Computer Organisation and Architecture, Palgrave 1996
Chu, Yaohan, Digital Computer Design Fundamentals, New York, San Francisco, Toronto, London 1962 (19h kyb 325 ec/833)
Chu, Yaohan, Introduction to Computer Organization, Englewood Cliffs, N. J. 1970
Chu, Yaohan, Computer Organization and Microprogramming, Englewood Cliffs, N. J. 1970, 1972
Chu, Yaohan (Hrsg.), High-Level Language Computer Architecture, New York , London. 1975
Cin, Mario Dal, Rechnerarchitektur : Grundzüge des Aufbaus und der Organisation von Rechnerhardware (Leitfäden der Informatik), Stuttgart 1996
Clippinger, Richard F., Programming Implications of Hardware Trends, in: Information Processing 1965, Proceedings of IFIP Congress '65, 2 Bde. Washington, D. C., London 1965, Bd. 1, S. 207-212
Cocke, John; Kolsky, Harwood G., The Virtual Memory in the Stretch Computer, in: AFIPS 16, Proceedings of the IRE/AIEE/ACM EJCC, (1959), S. 82-93
Computer Architecture News, Special Interest Committee on Computer Architecture, SICARCH, Association for Computing Machinery, ACM, New York, NY, 1.1972 -
Coy, Wolfgang, Aufbau und Arbeitsweise von Rechenanlagen, 2. Aufl. Wiesbaden 1992
Cragon, Harvey, Computer Architecture and Implementation, Cambridge, MA, London 2000
Daele, Wolfgang von den; Krohn, Wolfgang, Die Verwissenschaftlichung von Technologie, in: Böhme, G. u.a. (Hg.), Die gesellschaftliche Orientierung des wissenschaftlichen Fortschritts, Frankfurt a. M. 1978, S. 368-377
DalCin, Mario, Rechnerarchitektur: Grundzüge des Aufbaus und der Organisation von Rechnerhardware, Stuttgart 1996
Dasgupta, Subrata, The Design and Description of Computer Architectures, New York, Chichester, Brisbane 1984
Dasgupta, Subrata, Computer Architecture: A Modern Synthesis, Bd. 2: Advanced Topics, New York, Chichester, Brisbane 1989
Dasgupta, Subrata, A Hierarchical Taxonomic System for Computer Architecture, in: Computer 23 (1990) 3, S. 64-74
Dasgupta, Subrata, Creativity in Invention and Design: Computational and Cognitive Explorations of Technological Originality. New York: Cambridge University Press, 1994
Davies, P. M., Readings in Microprogramming, in: IBM Systems Journal 11 (1972) 1, S. 16-40
Deans, J. E.; Ramamoorthy, C. V., An Approach to Hardware/Software Architectural Trade-off Decisions, in: Fox, Jerome (Hrsg.), Computers and Automata, New York Apr. 1971, Brooklyn 1971, S. 193-206
De Blasi, Mario, Computer Architecture, Wokingham u.a. 1990
Denning, Peter J., Third Generation Computer Systems, in: Computing Surveys 3 (1971), S. 175-216
Dennis, Jack B., Modularity, in: Friedrich L. Bauer (Hrsg.), Software Engineering. An Advanced Course (Lecture Notes in Computer Science, Bd 30), Reprint Berlin, Heidelberg, New York 1975, S. 128-138
Dennis, Jack B., Programming Generality, Parallelism and Computer Architecture, in: Information Processing 1968, Proceedings of IFIP Congress '68, S. 484-492
Dennis, Jack B.; Fuller, S. H.; Ackerman, W. B.; Swan, R. J.; Weng, K.-S., Research Directions in Computer Architecture, in: Peter Wegner (Hrsg.), Research Directions in Software Technology, Cambridge, MA 1979, S. 514-556 (a kyb 400 h/332)
Despain, Alvin M., Notes on Computer Architecture for High Performance, in: Tiberghien, Jacques (Hrsg.), New Computer Architectures, London, Orlando, San Diego 1984, S. 59-138
Dirlewanger, Werner; Falkenberg, E. u.a., Einführung in Teilgebiete der Informatik, 2 Bde., Berlin, New York 1972-74
Dirlewanger, Werner; Hieber Ludwig; Rzehak, Helmut, Aufbau von Datenverarbeitungsanlagen, Berlin, New York 1976
Dotterweich, Dolores, Zuse-Dokumentation. Eine dokumentarische Auswertung der persönlichen Unterlagen von Prof. Dr. Zuse, in: Gebhardt, Friedrich (Hrsg.), Skizzen aus den Anfängen der Datenverarbeitung, GMD-Bericht Nr. 143, München, Wien 1983, S. 91-

Carpenter, B. E.; Doran, R. W., The other Turing Machine, in: Computer Journal 20 (1977) 3, S. 269-279
Doran, R. W., The Architecture of Stack Machines, in: Chu, Yaohan (Hrsg.), High-Level Language Computer Architecture, New York , London. 1975, S. 63-109
Doran, R. W., Computer Architecture: A Structured Approach, London, New York, San Francisco 1979 (A kyb 300 f/166)
Dunwell, Steve W., Design Objectives for the IBM STRETCH Computer, in: AFIPS, Bd. 8 (EJCC 1956), S. 20-22
Eadie, D., Data Processors and systems, Englewood Cliffs, N. J. 1971
East, Ian, Computer Architecture and Organization, London 1989
Englander, Irv, The Architecture of Computer Hardware and Systems Software : An Information Technology Approach, John Wiley & Sons 1996; 2. Aufl. 2000
Eckert, John Presper, Thoughts on the History of Computing, in: Computer (1976) 12, S. 58-65
Eckert, John Presper, What Hath He Wrought? in: Datamation 37 (1991) 3, S. 30-31
Eckert, John Presper, Report on the Magnetic Calculating Machine, 1944; Auszüge wiedergedruckt in: Stern, Nancy, From ENIAC to UNIVAC: An Appraisal of the Eckert-Mauchley Computers, Bedfors, Mass. 1981, S. 28, 75
Eckert, John Presper, Disclosure of Magnetic Calculating Machine, 29. Januar 1944, Kopie 1. Feb. 1945; gedruckt in: Lukoff, Herman From Dits to Bits: A Personal History of the Electronic Computer, Portland, OR. 1979, S. 207-209; wiedergedruckt in: Eckert, John Presper, The ENIAC, in: Metropolis, Nicholas C.; Howlett, Jack; Rota, Gian-Carlo (Hrsg.), A History of Computing in the Twentieth Century. A Collection of Essays, New York, London 1980, S. 537-539
Eckert, John Presper, A Preview of a Digital Computing Machine, Lecture 10, in: Campbell-Kelly, Martin; Williams, Michael R. (Hrsg.), The Moore School Lectures. Theory and Techniques for Design of Electronic Digital Computers (Charles Babbage Institute (Hrsg), Reprint Series for the History of Computing, Bd. 9), London, Los Angeles, San Francisco 1985, S. 109-126
Eckert, John Presper, UNIVAC-LARC. The Next Step in Computer Design, in: AFIPS 10, Proceedings of the IRE/AIEE/ACM EJCC (Dez. 1956), S. S.16-20
Eckert, John Presper, Die Elektronenrechner der Zukunft, in: ADL-Nachrichten (1962) 22, S. 110-112
Eckert, John Presper, In the Beginning and to What End, in: Williams, Richard H. (Hrsg.), World Computer Pioneer Conference, Computers and their Future. Speeches given at the World Computer Pioneer Conference Llandudno July, 1970, also questions and answers, Llandudno, UK 1970 (H kyb 300 t/831) S. 3/4-24
Eckert, John Presper, Thoughts on the History of Computing, in: Computer (1976) 12, S. 58-65
Eckert, John Presper, The ENIAC, in: Metropolis, Nicholas C.; Howlett, Jack; Rota, Gian-Carlo (Hrsg.), A History of Computing in the Twentieth Century. A Collection of Essays, New York, London 1980, S. 525-539
Eckert, John Presper, "What Hath He Wrought? in: Datamation 37 (1991) 3, S. 30-31
Eckert, John Presper; Chu, J. C.; Tonik, A. B.; Schmitt, W. F., Design of UNIVAC-LARC System I, in: AFIPS 16 (Dez. 1959) EJCC, S. 59-65
Eckert, John Presper; Lukoff, Herman; Smoliar, Gerry (?), A Dynamically Regenerated Electrostatic Memory System, in: Proceedings of the IRE 38 (1950), S. 498-510
Eckert, John Presper; Mauchly, John W., Automatic High-Speed Computing: A Progress Report on the EDVAC. Report No. W-670-ORD-4926, Supplement No. 4, Moore School Library, University of Pennsylvania, Philadelphia, 30. September 1945, University of Pennsylvania, Philadelphia 1946
Eckert, John Presper; Mauchly, John W.; Goldstine, Herman H.; Brainerd, John Grist, Description on the ENIAC, Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia Nov. 1945
Eckert, John Presper; Weiner, James B.; Welsh, H. Frazer; Mitchell, Herbert F., The UNIVAC System, in: AIEE-IRE Conference (Dez. 1951), S. 6-16; wiedergedruckt in: Bell, C. Gordon; Newell, Allen, Computer Structures: Readings and Examples, New York 1971, S. 157-169
Elliott, W. S., The Present Position of Automatic Computing Machine Development in England, in: Computation Laboratory of Harvard University (Hg.), Proc. of a Second Symposium on Large-Scale Digital Calculating Machinery Sept 1949, Cambridge, MA 1951, S. 74-80

ERA, Staff of Engineering Research Associates, Inc., High-Speed Computing Devices, New York 1950; in: Charles Babbage Institute (Hg.), Reprint Series for the History of Computing, Bd. 4, London u.a. 1983
Erhard, Werner, Rechnerarchitektur, Einführung und Grundlagen, Stuttgart 1995
Estrin, G., Organization of Computer Systems. The Fixed-Plus-Variable Strucure Computer, in: AFIPS 17 (1960) WJCC, S. 33-37
Evans, Bob O., SPREAD Report: The Origin of the IBM Sytem /360 Project , in: Annals of the History of Computing 5 (1983) 1, S. 4-44
Evans, Bob O., System /360: A Retrospective View, in: Annals of the History of Computing 8 (1986), S. 155-179
Everett, Robert R., The Whirlwind I Computer (1952), wiedergedr. in: Bell, Newell, 1971, S. 137-145
Everling, Wolfgang, Rechnerarchitekturen (Reihe Informatik ; 72), Mannheim u.a. 1991
Everling, Wolfgang, Rechnerorganisation, Heidelberg, Berlin, Oxford 1993
Everling, Wolfgang, Algebra der Rechnerarchitekturen, Heidelberg, Berlin, Oxford 1996
Fagg, P. J.; Brown, J. A. u. a., IBM System /360 Engineering, in: AFIPS, Bd. 26 (FJCC 1964), S. 205-231
Flores, Ivan, Computer Logic. The Functional Design of Digital Computers, Englewood Cliffs, N. J. 1960 (A kyb 327 e 487)
Flores, Ivan, Computer Design, Englewood Cliffs, N. J. 1967 (A kyb 326 e 691)
Flynn, Michael J., Very High-Speed Computing Systems, in: Proceedings of the IEEE 54 (1966), S. 1901-1909
Flynn, Michael J.; Low, R. P, The IBM System /360 Model 91: Some Remarks on System Development, in: IBM Research and Development 11 (1967) 1, S. 2-7
Flynn, Michael J., Some Computer Organizations and Their Effectiveness, in: IEEE Transactions on Computers C-21-9, 21 (Sept 1972), S. 948-960
Flynn, Michael J., Toward More Efficient Computer Optimizations, in: in: AFIPS, Bd. 40 (SJCC 1972), S. 1211-1217
Flynn, Michael J., Microprogramming - Another Look at Internal Computer Control, in: Proceedings of the IEEE 63 (1975) 11, S. 1554-1567
Flynn, Michael J., Directions and Issues in Architecture and Language, in: Computer 13 (1980) 10, S. 5-22
Flynn, Michael J., Computer architecture: pipelined and parallel processor design, Boston u.a. 1995
Forrester, Jay W., The Digital Computation Program at Massachusetts Institute of Technology, in: Computation Laboratory of Harvard University (Hrsg.), Proc. of a Second Symposium on Large-Scale Digital Calculating Machinery Sept 1949, Cambridge, MA 1951, S. 44-49
Forrester, J. W., Digital Computers:Past and Present, in : Review of Electronic Digital Computers.Joint AIEE-IRE Computer Conference Dez.1951, New York 1952, S.109-114
Foster, Caxton C., A View of Computer Architecture, in: Communications of the ACM, 15 (1972), 7, S. 557-565
Foster, Caxton C., Computer Architecture, New York 1970
Foster, Caxton C., The Next Three Generations, in: Computer 5 (1972) 3/4, S. 39-42
Foster, Caxton C., Computer Architecture, in: Ralston, Antony; Meek, Chester L. (Hrsg.), Encyclopedia of Computer Science, New York 1976, S. 263-268
Freeman, Herbert (Hrsg.), Machine Vision : Algorithms, Architectures, and Systems, Machine Visionalgorithms, Architectures, and Systems (Perspectives in Computing ; 20) Boston 1988
Freeman, P., The Context of Design, in: Freeman, P.; Wassermann, A. (Hrsg.), Tutorial on Software Design Techniques, 3. Aufl. New York 1980, S.
Friedman, Andrew L.; Cornford, Dominic S., Computer Systems Development. History, Organization, and Implementation, Chichester, New York, Brisbane 1989, Reprint 1993
Fuller, Samuel H.; Burr, W. E., Measurement and Evaluation of Alternative Computer Architectures, in: Computer 10 (1977) 10, S. 24-35
Fuller, Samuel H.; Siewiorek, Daniel P., Some Observations on Semiconductor Technology and the Architecture of Larger Digital Modules, in: IEEE Computer 6 (1973) 10, 14-21
Furger, Franco; Heintz, Bettina, Technologische Paradigmen und lokaler Kontext. Das Beispiel der ERMETH, in: Schweizerische Zeitschrift für Soziologie 23 (1997) 3, S. 533-566
Furger, Franco; Heintz, Bettina, Wahlfreiheiten. Frühe Computerentwicklung am Beispiel der Schweiz, in: Siefkes, Dirk; Eulenhöfer, Peter; Stach, Heike; Städtler, Klaus (Hrsg.), Sozialgeschichte der Informatik. Kulturelle Praktiken und Orientierungen, Wiesbaden 1998, S. 231-253

Ganzhorn, Karl, Prinzipien in Rechnerstrukturen, in: Hasselmeier, H.; Spruth, W. G., Rechnerstrukturen. Vorträge des Informatik-Symposiums der IBM Deutschland, Wildbad 1973, München, Wien 1974, S. 434-450

Ganzhorn, Karl E., Schwellen des Fortschritts in der Informationstechnik, in: Methner, H.; Oberhoff, W. D.; Schmidt, P.; Swiridow, A. P.; Unger, H.; Vollmar, R. (Hrsg.), Computer und Kybernetik. Anmerkungen zu ihrer Geschichte und zu Perspektiven in der Zeit von 1940 bis 1965. 3. Russisch-Deutsches Symposium, Heidelberg November 1997; S. 45-53

Ganzhorn, Karl E.; Schulz, K. M.; Walter, W. (1981), Datenverarbeitungssysteme, Berlin, Heidelberg, New York

Garside, R. G., The Architecture of Digital Computers, Oxford 1980 (A kyb 326 e/241)

Gear, C. W., Computer Organization and Programming, 2. Aufl. New York u. a. 1974

Giloi, Wolfgang K., Grundlagen, Operationsprinzipien und Strukturen von innovativen Rechner-architekturen, in: GI-8. Jahrestagung, (Informatik-Fachberichte 16), Berlin, Heidelberg, New York 1978, S. 274-307

Giloi, Wolfgang K., Rechnerarchitektur, in: Informatik Spektrum 3 (1980) 1, S. 3-18

Giloi, Wolfgang K., Rechnerarchitektur, 1. Aufl. Berlin, Heidelberg, New York, 1981; 2. Aufl. 1993; 3. Aufl. (fraglich)

Giloi, Wolfgang K., A Complete Taxonomy of Computer Architectures Based on the Abstract Data Type View, in: IFIP Workshop on Taxonomy in Computer Architecture, Erlangen 1981, S. 19-38

Giloi, Wolfgang K., Rechnerarchitektur - heute und morgen. GI Jahrestagung 1982, S. 1-29

Giloi, Wolfgang K., Towards a Taxonomy of Computer Architecture Based on the Machine Data Type View, in: Proceedings Tenth Annual International Symposium on Computer Architecture, IEEE, Stockholm, Juni 1983, Los Alamitos, Cal. 1983, S. 6-15

Giloi, Wolfgang K., Die Entwicklung der Rechnerarchitektur von der von-Neumann-Maschine bis zu den Rechnern der fünften Generation, in: Elektronische Rechenanlagen 26 (1984) 2, S. 55-70

Goldberg, Robert F., Architecture of Virtual Machines, in: AFIPS 42 (1973) Proceedings of NCC, S. 309-318

Goldberg, Robert F., Survey of Virtual Machines Research, in: Computer 7 (1974) 6, S. 34-43

Goldstine, Adele K., Report on the ENIAC (Electronic Numerical Integrator and Computer), Technical Report 1, 2. Bde., Philadelphia, 1.Juni 1946

Goldstine, Herman H., A Report on the ENIAC, Part I, Bd. I und II, Technical Description on the ENIAC, Moore School of Electrical Engineering, University of Pennsylvania , Philadelphia 1946

Goldstine, Herman H., Some Experience in Coding and Programming with the Institute Computer, in: Symposium on Large Scale Digitla Computing Machinery, Proceedings, US Atomic Energy Commission 1953

Goldstine, Herman H., Some Remarks on Logical Design and Programming Checks, in: AFIPS Bd. 4 (1953) EJCC, S. 96-98

Goldstine, Herman H., Systematics of Automatic Electronic Computers, in: Hoffmann, Walter; Walther, Alwin (Hrsg.), Elektronische Rechenmaschinen und Informationsverarbeitung (Nachrichtentechnische Fachberichte, Bd. 4), Braunschweig 1956, S. 1-4

Goldstine, Herman H., On the Relation between Machine Developments and Numerical Analysis, in: Bonner Mathematische Schriften 2/3 (1957), S. 1-7

Goldstine, Herman H., Interrelations between Computers and Applied Mathematics, in: Hoffmann, Walter (Hrsg.), Digitale Informationswandler. Probleme der Informationsverarbeitung in ausgewählten Beiträgen, Braunschweig 1962, S.

Goldstine, Herman H., Early Electronic Computers, in: Fernbach, S.; Taub, A. H., Computers and their Role in the Physical Sciences, New York 1970, S. 51-102

Goldstine, Herman H., The Computer from Pascal to von Neumann, Princeton, N. J. 1972 (19h kyb 007/823)

Goldstine, Herman H., Early Electronic Computers, in: S. Fernbach; Taub, A., Computers and Their Role in the Physical Sciences, New York 1970, S.

Goldstine, Herman H., A History of Numerical Analysis from the 16th through the 19th Century, (Studies in the history of mathematics and physical sciences ; 2), New York, Berlin, Heidelberg 1977 (A mat 006.5/190)

Goldstine, Herman H., A history of the calculus of variations from the 17th through the 19th century (Studies in the history of mathematics and physical sciences ; 5), New York, Berlin, Heidelberg 1980 (A mat 006.5/290)

Goldstine, Herman H., Remembrance of Things Past, in: Nash, Stephen G. (Hrsg.), A History of Scientific Computing, Reading, MA 1990, S. 5-19

Goldstine, Herman H.; Goldstine, Adele, The Electronic Numerical Integrator and Computer (ENIAC), in: Mathematical Tables and other Aids to Computation 2 (1946) Juli, S. 97-110; wiedergedruckt in: Randell, Brian (Hrsg.), The Origins of Digital Computers. Selected Papers, Berlin, Heidelberg, New York 1973, S. 333-347; 2. Aufl. 1975; 3. Aufl. 1982, S. 359-373; ebenso in: Annals of the History of Computing 18 (1996) 1, S. 10-16

Goldstine, Herman Heine; Neumann, John von, Planning and Coding of Problems for an Electronic Computing Instrument, 3 Bde. Institute for Advanced Study, Princeton 1947-1948; wiedergedruckt in: Taub, A. H. (Hrsg.), Complete Works of John von Neumann, 6 Bde.Oxford (UK), New York 1961-63, Bd. 5, Teil I:S. 80-151; Teil II: S. 152-214; Teil III: S. 215-235; und in: Aspray, William; Burks, Arthur W. (Hrsg.), Papers of John von Neumann on Computing and Computing Theory, Cambridge/ Mass., London, Los Angeles, San Francisco 1987, Teil I:S. 151-222; Teil II: S. 223-285; Teil III: S. 286-306

Goldstine, Herman Heine; Neumann, John von, On the Principles of Large Scale Computing Machines (Ms. 1946), gedruckt in: Taub, Abraham H. (Hrsg.), Complete Works of John von Neumann, 6 Bde.Oxford (UK), New York 1961-63, Bd. 5, S. 1-34

Goldstine, Herman Heine; Wigner, Eugene P., Scientific Work of J. von Neumann, in: Science 125 (1957), S. 683-684

Goode, H. H.; Machol, R., E., System Engineering, New York u.a. 1957

Goor, A. J. van de, Computer Architecture and Design, Wokenham u.a. 1989

Gray, Harry J., Digital Computer Engineering, Englewood Cliffs, N. J. 1963 (A kyb ei 559)

Gschwind, Hans W., Design of Digital Computers. An Introduction, New York, Wien 1967 (kyb 326e244)

Gutknecht, Jürg (Hrsg.), Programming Languages and System Architectures, International Conference, Zürich, March 2 - 4, 1994 , Proceedings (Lecture notes in Computer Science ; 782), Berlin, Heidelberg, New York 1994

Haanstra, J. W.; Evans, Bob O.; Aron, Joel D.; Brooks, Frederick P. u. a. , Processor Products - Final Report of SPREAD Task Group, reprinted in: Annals of the History of Computing 5 (1983) 1, S. 6-26

Händler, Wolfgang (Hrsg.), Computer Architecture. Workshop of the Gesellschaft für Informatik,Erlangen, Mai 1975 (Informatik-Fachberichte 4), Berlin, Heidelberg, New York 1976

Händler, Wolfgang, The Impact of Classification Schemes on Computer Architecture, in: B. Shaw (Hrsg.), Digital Systems Design. Proceedings of the Joint IBM University of Newcastle upon Tyne Seminar, Sept. 1977, Newcastle 1978, S. 89-101

Händler, Wolfgang, Standards, Classification, and Taxonomy: Experiences with ECS, in: Blaauw, Gerrit A.; Händler, Wolfgang (Hrsg.), Workshop on Taxonomy in Computer Architecture, Universität Erlangen-Nürnberg 1981, S. 39-75

Händler, Wolfgang; Haupt, Dieter; Jeltsch, ; Juling, Wilfried; Lange, Otto (Hrsg.), CONPAR 86, Conference on Algorithms and Hardware for Parallel Processing, Proceedings Aachen, September 1986 (Lecture Notes in Computer Science), Berlin, Heidelberg, New York 1986

Haley, A. C. D., DEUCE: A High-speed General-Purpose Computer, in: Proceedings of the IEE 103 B, Suppl. 1-3 (1956), S. 165-173

Hall, Arthur David, III, A Methodology for Systems Engineering, New York 1962

Harland, David M., REKURSIV: Object-oriented Computer Architecture, Chichester u.a., 1988

Hartree, Douglas R., Recent Developments in Calculating Machines, in: Journal of Scientific Instruments, 24 (1947), S. 172-176

Hartree, Douglas R., Calculating Machines. Recent and Prospective Developments and Their Impact on Mathematical Physics, London, New York 1947; Reprint Cambridge, Mass., London 1984; und in: Charles Babbage Institute (Hrsg), Reprint Series for the History of Computing, Bd. 6, London, Los Angeles, San Francisco 1984

Hartree, Douglas R., Calculating Instruments and Machines, Cambridge University Press 1949, 1950 Urbana, Ill. 1949; Reprint in: Charles Babbage Institute (Hrsg), Reprint Series for the History of Computing, Bd. 6, London, Los Angeles, San Francisco 1984

Hartree, Douglas R., Calculating Instruments and Machines. Recent and Prospective Developments and Their Impact on Mathematical Physics *and* Calculating Instruments and Machines, with a New introduction by Maurice Wilkes (Charles Babbage Institute (Hrsg), Reprint Series for the History of Computing, Bd. 6), London, Los Angeles, San Francisco 1984

Hartree, Douglas R., General Introduction, in: Report of a Conference on High Speed Automatic Calculating Machines 1949; wiedergedruckt in: The Early British Computer Conferences edited and introduction by Michael R. Williams and Martin Campbell-Kelly

Charles Babbage Institute (Hrsg), Reprint Series for the History of Computing, Bd. 14, London, Los Angeles, San Francisco 1989, S. 12-15
Hartree, Douglas R., Modern Calculating Machines", Endeavour 1949
Hartree, Douglas R., Automatic Calculating Machines, in: The Mathematical Gazette 34 (310) (Dez.1950), S. 241-252
Hartree, Douglas R., Automatic Digital Computation. Opening Adress, in: International Symposium on Automatic Digital Computation (National Physical Laboratory, 1953), London 1954; wiedergedruckt in: The Early British Computer Conferences edited and introduction by Michael R. Williams and Martin Campbell-Kelly Charles Babbage Institute (Hrsg), Reprint Series for the History of Computing, Bd. 14, London, Los Angeles, San Francisco 1989, S. 215-218
Hasselmeier, H.; Spruth, W. G., Rechnerstrukturen. Vorträge des Informatik-Symposiums der IBM Deutschland, Wildbad 1973, München, Wien 1974
Hayes, John P., Computer Architecture and Organization, New York, St. Louis, San Francisco 1978; 2. Aufl. New York, St. Louis, San Francisco 1988 (Kap. 1= The Evolution of Computers, S. 1-84); 3. Aufl. Boston u.a. 1998
Heath, Frederick George, Digital Computer Design (Electronic and Electrical Engineering Texts Bd. 3), Edinburgh 1969 (19h kyb 326/222)
Heath, Frederick George, Specification and Evaluation of Computer Structures, in: Boulaye, Guy G.; Lewin, Douglas W. (Hrsg.), Computer Architecture. Proceedings of the NATO Advanced Study Institute, St. Raphael, Sept. 1976 (NATO Advanced Study Institutes Series, Serie C, Bd. 32), Dordrecht, Boston 1977, S. 141-170
Hellerman, Herbert, Digital Computer System Principles, New York 1967; 2. Aufl. 1972
Hellige, Hans Dieter, Leitbilder und historisch-gesellschaftlicher Kontext der frühen wissenschaftlichen Konstruktionsmethodik, artec-Paper Nr. 8, Januar 1991, 53 S.
Hellige, Hans Dieter, Von der programmatischen zur empirischen Technikgeneseforschung, in: Technikgeschichte, Bd. 60 (1993), Nr. 3, S. 186-223
Hellige, Hans Dieter, Vom thermodynamischen Kreisprozeß zum recyclinggerechten Konstruieren, in: W. Müller (Hg.), Der ökologische Umbau der Industrie, Münster, Hamburg 1995, S. 73-109
Hellige, Hans Dieter, Hierarchische Ablaufsteuerung oder kooperative Bewältigung von Problemzusammenhängen? Zur Geschichte von Modellen des Konstruktionsprozesses, in: H. Lange, W.Müller (Hg.), Kooperation in der Arbeits- und Technikgestaltung, Münster, Hamburg 1995, S. 135-164
Hennessey, John L., The Future of Systems Research, in: IEEE Computer 32 (1999) 8, S. 27-33
Hennessey, John L.; Jouppi, Norman P., Computer Technology and Architecture: an Evolving Interaction, in: IEEE Computer 24 (1991) 9, S. 18-29
Hennessey, John L., The Future of Systems Research, in: IEEE Computer 32 (1999) 8, S. 27-33
Hennessey, John L.; Patterson, David A., Computer Architecture: A Quantitative Approach, San Mateo, CA 1990; 2. Aufl. San Francisco u.a. 1996
Hennessey, John L.u. a., Hardware/Software Tradeoffs for Increased Performance, in: Proceedings of the Symposium on Arch. Support for Programming, Languages and Operation Systems 10 (1982) 2, S. 2-11
Hennessy, John L.; Patterson, David A., Computer Organization and Design : The Hardware/ Software Interface, San Mateo, CA 1994 (BB Math.-MZH:19h inf 110/848)
Hennessy, John L.; Patterson, David A., Rechnerarchitektur : Analyse, Entwurf, Implementierung, Bewertung, Braunschweig, Wiesbaden 1994.(BB Math.-MZH:19h inf 210 ec/209); 2. Aufl. San Francisco 1998
Herken, Rolf (Hrsg.), The Universal Turing Machine. A Half-Century Survey, (Computerkultur, Bd. II), 2. Aufl. Wien, New York 1995
Herrmann, Paul, Rechnerarchitektur. Aufbau, Organisation und Implementierung, Braunschweig, Wiesbaden 1998
Heuring, Vincent P.; Jordan, Harry F., Computer Systems Design and Architecture, Benjamin Cumming 199
Hill, Mark D., Readings in Computer Architecture, San Francisco, CA 2000
Hill, R. J.; Peterson, G. R., Digital Systems: Hardware Organization and Design, New York 1973
Hintze, Guenther, Fundamentals of Digital Machine Computing, Berlin, Heidelberg, New York 1966
Hoffmann, Walter (Hrsg.), Digitale Informationswandler. Probleme der Informationsverarbeitung in ausgewählten Beiträgen, Braunschweig 1962

Hohn, Hans-Willy, Kognitive Strukturen und Steuerungsprobleme der Forschung: Kernphysik und Informatik im Vergleich (Schriften des Max-Planck-Instituts für Gesellschaftsforschung Köln, Bd. 36), Frankfurt a. M., New York 1998 (01 G 2720)
Holtkamp, Bernhard, Angepaßte Rechnerarchitektur, Stuttgart 1986
Hopper, Grace Murray; Mauchly, John W., Influence of Programming Techniques on the Design of Computers, in: Proceedings of the IRE 41 (1953) 10, S. 1250-1254
Howarth, D. J.; Payne, R. B.; Sumner, F. H., The Manchester University Atlas Operating System, Part II: User's Description, in: Computer Journal 4 (1961) 3, S. 226-229
Hurd, Cuthbert C., Computer Development at IBM, in: Metropolis, Nicholas C.; Howlett, Jack; Rota, Gian-Carlo (Hrsg.), A History of Computing in the Twentieth Century. A Collection of Essays, New York, London 1980, S. 389-418
Hurd, Cuthbert C., Early IBM Computers: Edited Testimony, in: Annals of the History of Computing 3. (1981) 1., S. 163-182
Husson, S. S., Microprogramming: Principles, and Practices, Englewood Cliffs, N. J. 1970
Hwang, Kai; Briggs, Fayé A., Computer Architecture and Parallel Processing, New York, St. Louis, San Francisco 1984
Hwang, Kai, Advanced Computer Architecture. Parallelism, Scalability, Programmability, New York, St. Louis, San Francisco 1993
Hwang, Kai, Parallelverarbeitung, 1994
IBM System Development Division, System Networks Architecture. General Information, Research Triangle Park, N. C. 1975
Irwin, Wayne C., Digital Computer Principles, Princeton, N.J, Toronto, London, New York 1960
Jensen, E. Douglas, The Influence of Microprocessors on Computer Architecture: Distributed Processing, in: Proceedings of the ACM National Conference 1975, S. 125-128
Jessen, Eike, Architektur digitaler Rechenanlagen, Berlin, Heidelberg, New York 1975
Johnson, Lyle R., Architecture, in: Annals Hist Comput 22 (2000) 2, S.70-71
Johnson, Robert R., Perspectives on Computers: What They Are and What They Do, in: Leilich, H.-O. (Hrsg.), Fachtagung Struktur und Betrieb von Rechensystemen, Braunschweig März 1974 (Computer Science, Bd. 8), Berlin, Heidelberg, New York 1974
Jones, P. D., Operating System Structures, in: Information Processing 1968, Proceedings of IFIP Congress '68, S. C 29-33
Jones, P. D.; Lincoln, N. R.; Thornton, J. F., Whither Computer Architecture? in: Information Processing '71, Proceedings of IFIP Congress 1971, Amsterdam 1972, S. 729-736
Joseph, Earl C., Computers: Trends Toward the Future, in: Information Processing 68, Amsterdam 1969, S. 665-679
Joseph, Earl C., Evolving Digital Computer System Architectures, in: IEEE Computer Group News 2 (1969) 8, S. 2-8
Joseph, Earl C., Future Computer Architectures - Polysystems, in: COMPCOM '72, Digest of Papers, IEEE Computer Society 1972, S. 149-154
Jungmann, Dieter; Stange, Horst, Einführung in die Rechnerarchitektur, München, Wien 1992
Kain, Richard Y, Computer Architecture: Software and Hardware, Englewood Cliffs, N.J. 1989
Kain, Richard Y, Advanced Computer Architecture : A Systems Design Approach, Englewood Cliffs, N. J. u.a. 1996
Karalis, Edward Digital design principles and computer architecture, Upper Saddle River, N.J. u.a. 1997
Katzan, Harry, Computer Organization and the System/370, New York 1971
Kline, Raymond M., Digital Computer Design, Englewood Cliffs, N.J. 1977 (A kyb 326 e/544)
König, Wolfgang, Elektrotechnik – Entstehung einer Industriewissenschaft, Berlin 1993
König, Wolfgang, Elektrotechnik – Entstehung einer wissenschaftlichen Disziplin, Berichte zur Wissenschaftsgeschichte 10 (1987), S. 83-93
König, Wolfgang, Künstler und Strichezieher, Frankfurt a. M. 1999
Kogge, P. M. The Architecture of Pipelined Computers, New York 1981
Kroneberg, D., What is Systems Architecture? in: Proceedings of the 1st European Workshop on Computer Networks, Arles, April/Mai 1973, S. 223-225
Kuck, David J., The Structure of Computers and Computations, Bd. 1, New York, Santa Barbara, Chichester 1978
Lampson, Butler W.; Paul, M.; Siegert, H. J. (Hrsg.), Distributed Systems. Architecture and Implementation, Berlin, Heidelberg, New York 1981
Landauer, R., The Future Evolution of the Computer, in: Physics Today 22 (Jul 1970), S. 22
Langdon, Jr., Glen G., Logic Design. A review of theory and practice, New York, San Francisco, London, 1974
Langefors, B.; Sundgren, B., Informations Systems Architecture, New York 1975

Lavington, Simon H., Manchester Computer Architectures, 1948-1975, in: Annals of the History of Computing 15 (1993) 3, S.44-54
Lawless, W. J. Jr., Developments in Computer Logical Organzation, in: Advances in Electronics 10 (1959), S. 153-184
Lawson, Harold W., Jr., Computer Architecture Education, in: Tiberghien, Jacques (Hrsg.), New Computer Architectures, London, Orlando, San Diego 1984, S. 225-286
Ledley, Robert Steven, Digital Computer and Control Engineering, New York 1960 (HB=00)
Lewin, Douglas, Theory of Digital Computer Systems, 2. Aufl. Walton-on-Thames 1980 (A kyb 325e/433-2)
Lichtenberger, W. W.; Pirtle, M. W., A Facility for Experimentation in Man-Machine Interaction, in: AFIPS Bd. 27 (FJCC) 1965, S. 589-598
Lincoln, N. R., Technology and Design Trade Offs in the Creation of a Modern Supercomputer, in: Transactions on Computer, C-31, 31 (1982) 5, S. 363-376
Lipovski, G. Jack; Doty, Keith L., Developments and Directions in Computer Architecture, in: IEEE Computer, Aug. 1978, S. 54-67
Lorin, H., Parallism in Hardware and Software: Real and Apparent Concurrency, Englewood Cliffs, N. J. 1972
Lukoff, Herman, Were Early Giant Computers a Success? in: Datamation (1969) 4, S. 77-82
Lukoff, Herman, From Dits to Bits: A Personal History of the Electronic Computer, Portland, OR. 1979
Lukoff, Herman; Spandorfer, L. M.; Lee, F. F., Design of UNIVAC-LARC System II, in: AFIPS 16 (Dez. 1959) EJCC, S. 59-65
Madnick, S. E.; Donovan, J. J., Operating Systems, McGraw Hill 1974
Märtin, Christian, Rechnerarchitektur: Struktur, Organisation, Implementierungstechnik, München, Wien 1994
Märtin, Christian, Rechnerarchitektur: CPUs, Systeme, Software-Schnittstellen, Leipzig 2001
Manheim, Marvin L., Hierarchical Structure: A Model of Planning and Design Processes, Cambridge, MA 1966
Mano, M. Morris, Computer Logic Design, Englewood Cliffs, N.J. 1972
Mano, M. Morris, Computer System Architecture, Englewood Cliffs, N.J. 1976 (kyb 326 e 057), 2. Aufl. Englewood Cliffs, N. J. 1982; 3. Aufl. Englewood Cliffs, N.J. 1993
Mano, M. Morris, Computer Engineering Hardware Design, Englewood Cliffs, N.J. 1988
Marcus, Mitchell; Akera, Atsushi, Exploring the Architecture of an Early Machine: The Historical Relevance of the ENIAC Machine Architecture, in: Annals of the History of Computing 18 (1996) 1, S. 17-24
Mauchly, John W., The Use of High Speed Vacuum Tube Devices for Calculating (Aug. 1942), wiedergedruckt in: Randell, Brian (Hrsg.), The Origins of Digital Computers. Selected Papers, Berlin, Heidelberg, New York 1973, S.329-332; 2. Aufl. 1975; 3. Aufl. 1982, S. 355-358
Mauchly, John W., Digital and Analogy Computing Machines, Lecture 1, in: Campbell-Kelly, Martin; Williams, Michael R. (Hrsg.), The Moore School Lectures. Theory and Techniques for Design of Electronic Digital Computers (Charles Babbage Institute (Hrsg), Reprint Series for the History of Computing, Bd. 9), London, Los Angeles, San Francisco 1985, S. 25-40
Mauchly, John W., Amending the ENIAC Story, in: Datamation 25 (1979) 10, S. 217-224
Mauchly, John W., The ENIAC, in: Metropolis, Nicholas C.; Howlett, Jack; Rota, Gian-Carlo (Hrsg.), A History of Computing in the Twentieth Century. A Collection of Essays, New York, London 1980, S. 541-559
Mauchly, John W., Amending the ENIAC Story, in: Datamation (1979) 10, S. 217-220
Mauchly, John W.; Eckert, John Presper; Brainerd, John Grist, Report on an Electronic Diff. Analyzer, Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, 12. April 1943
Mauchly, Kathleen R., John Mauchly's Early Years , in: Annals of the History of Computing, 6 (1984) 2, S. 116-138
McKeeman, W. M., Language Directed Computer Design, in: AFIPS Bd. 31 (FJCC) 1967, S. 413-417
Mealy, George H., The System Design Cycle, in: Proceedings of the 2nd Symposium on Operating System Principles, Princeton University, Oktober 1969, S. 1-7
Mealy, George H.; Witt, B. I.; Clarke, W. A., The Functional Stucture of OS/360, in: IBM Systems Journal 5 (1966) 1, 2-51
Metropolis, Nicholas C.; Howlett, Jack; Rota, Gian-Carlo (Hrsg.), A History of Computing in the Twentieth Century. A Collection of Essays, New York, London 1980
Moon, D. L., Digital Machine Design and Analysis, in: Computer Design 9 (1970) 7, S. 59-65

Müller, Silvia Melitta; Paul, Wolfgang J., The Complexity of Simple Computer Architectures (Lecture Notes in Computer Science; 995), Berlin, Heidelberg, New York 1995
Müller, Silvia Melitta; Paul, Wolfgang J., Computer Architecture: Complexity and Correctness, Berlin, Heidelberg, New York 2000 (A inf 120 ris 680)
Murdocca, Miles J.; Heuring, Vincent P., Principles of Computer Architecture, Upper Saddle River, N.J. 2000
Murray, William D., Computer and Digital System Architecture, Englewood Cliffs, N.J. 1990
Myers, Glenford J., Advances in Computer Architecture, New York, Chichester, Brisbane 1978 Akyb 326 e 755); 2. Aufl. New York, Chichester, Brisbane 1982 ,
Myers, Glenford J., The Design of Computer Architectures to Enhance Software Reliability, Phil. Diss.Polytechnic Institute New York 1977
Negroponte, Nicholas, The Architecture Machine: Toward a More Human Environment, Cambridge, Mass. 1970
Netherwood, Douglas B., Logical Machine Design: A Selected Bibliography, Teil I, in: IRE-Transactions on Electronic Computers EC-7, 7 (1958) 2, S. 155-178; Teil II in: ebda., EC-8, 8 (1959) 3, S. 367-380
Neumann, John von, Collected Works, 1903-1957, hrsg. von Taub, A. H., 6 Bde.Oxford (UK) 1961-63
Neumann, John von, First Draft of a Report on the EDVAC, Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia 30. Juni 1945; wiedergedruckt in: Randell, Brian (Hrsg.), The Origins of Digital Computers. Selected Papers, Berlin, Heidelberg, New York 1973, S. 355-364 (gekürzt); 2. Aufl. 1975; 3. Aufl. 1982; wiedergedruckt in: William Aspray, Arthur W. Burks (Hrsg.), Papers of John von Neumann on Computing and Computing Theory, Cambridge, Mass., London, Los Angeles, San Francisco 1987, S. 17-82; korrigierte Fassung nach dem Originalmanuskript, hrsg. von Michael D. Godfrey in: Annals of the History of Computing 15 (1993) 4, S. 27-67
Neumann, John von, The Principles of Large-Scale Computing Machines (1946), wiedergedruckt in: Annals of the History of Computing 3 (1981) 3, S. 263-273 [kein Original-Ms, siehe Neupublikation 1989]
Neumann, John von, The Principles of Large-Scale Computing Machines (Mai 1946), wiedergedruckt in: Annals of the History of Computing 10 (1989) 4, S. 243-256
Neumann, John von, The Future Role of Rapid Computing in Meteorology, in: Aeronautical Engineering Review 6 (1947) 4, S. 30
Neumann, John von, The Future of High-Speed Computing, in: Proceedings, Computation Seminar, 5.-9. Dez. 1949, S. 13, IBM Corporation, New York 1949
Neumann, John von, The General and Logical Theory of Automata, in: Jeffres, Lloyd A. (Hrsg.), Cerebral Mechanisms in Behavior. The Hixon Symposium, New York 1951, S. 1-31; wiedergedruckt in: Pylyshyn, Zenon W., (Hrsg.), Perspectives On the Computer Revolution, Englewood Cliffs, N.J. 1970, S. 87-113
Neumann, John von, On a General and Logical Theory of Automata (1951), in: Aspray, William; Burks, Arthur (Hrsg.), Papers of John von Neumann on Computing and Computer Theory, Cambridge, MA, London, Los Angeles, San Francisco 1987, S. 391-431
Neumann, John von, Allgemeine und logische Theorie der Automaten (1951), in: Kursbuch 8 (1967), S. 139-175
Neumann, John von, The NORC and Problems in High-Speed Computing. Speech at first public showing of the IBM Naval Ornance Research Calculator 2. Dez.1954, gedruckt in: Annals of the History of Computing 3 (1981) 3, S. 274-279
Neumann, John von, Entwicklung und Ausnutzung neuerer mathematischer Maschinen, in: Arbeitsgemeinschaft für Forschung des Landes Nordrhein-Westfalen, Heft 45), Köln, Opladen 1955, S. 7-27 (45. Sitzung am 15. Sept. 1954 in Düsseldorf; Diskussion, S. 47-65); wiedergedruckt in: Schuchmann, Hans-Rainer; Zemanek, Heinz (Hrsg.), Computertechnik im Profil, München, Wien 1984, S.
Neumann, John von, The Computer and the Brain, New Haven 1958
Neumann, John von, Die Rechenmaschine und das Gehirn, München, Wien 1960; 2. Auflage 1965; 3. Aufl. 1970 (TB Technik:G III 1083)
Neumann, John von; Burks, Arthur W., Theory of Self-Reproducing Automata, edited and completed by Artur W. Burks, Urbana 1966
Neumann, Nicholas A., John von Neumann as Seen by His Brother, Meadowbrook,, PA 1987
Neumann, Peter C., Toward a Methodology for Designing Large Systems and Verifying Their Properties, in: GI-4. Jahrestagung, Berlin Okt. 1974 (Lecture Notes in Computer Science, Bd. 26), Berlin, Heidelberg, New York 1975, S. 52-66

Oberschelp, Walter; Vossen, Gottfried, Rechneraufbau und Rechnerstrukturen, 4. verb. Aufl. München, Wien 1990; 8. Aufl. München, Wien 2000
Oelsner, Reiner F., Geschichte des Konstruierens in Deutschland. Vom künstlerischen Handeln zum formalisierten Wissen, in: Konstruktion 44 (1992) 12, S. 387-390
Organick, Elliott I., Computer System Organization. The B5700/B6700 Series, New York 1973
Page-Jones; M., Strukturiertes Systemdesign, München, Wien 1994
Papadimtriou, Christos, Computational Complexity, Reading, Mass., Menlo Park, New York 1994, 2. Aufl. 1995
Parnas, David L.; Darringer, J. A., SODAS and a Methodology for System Design, in: AFIPS Bd. 31 (FJCC) 1967, S. 449-474
Patterson, David A., Reduced Instruction Set Computers, in: Communications of the ACM 28 (1985) 1, S. 8-21
Patterson, G. W., Theory and Techniques for Design of Electronic Digital Computers, The University of Pennsylvania, Moore School of Electrical Engineering, Report Nr. 47-21, 1947; wiedergedruckt in: Campbell-Kelley, Martin; Williams, Michael R. (Hrsg.), The Moore School Lectures, Cambridge, MA, Los Angeles 1985, S.
Pearson, Jamie Parker (Hrsg.), Digital at Work. Snapshots from the first thirty-five years, Burlington, MA 1992
Peschel, M.; Riedel, C., Polyoptimierung, eine Entscheidungshilfe für ingenieurtechnische Kompromißlösungen, Berlin 1976
Petzold, Hartmut, Rechnende Maschinen. Eine historische Untersuchung ihrer Herstellung und Anwendung vom Kaiserreich bis zur Bundesrepublik (Technikgeschichte in Einzeldarstellungen, Bd. 41), Düsseldorf 1985
Petzold, Hartmut, Moderne Rechenkünstler. Die Industrialisierung der Rechentechnik in Deutschland, München 1992
Petzold, Hartmut, Die Mühlen des Patentamts. Die vergeblichen Bemühungen Konrad Zuses, die programmgesteuerte Rechenmaschine patentieren zu lassen, in: Rojas, Raúl (Hrsg.), Die Rechenmaschinen von Konrad Zuse, Berlin, Heidelberg, New York 1998, S. 63-108
Petzold, Hartmut, Konrad Zuse and Industrial Manufacturing of Electronic Computers in Germany, in: Rojas, Raúl; Hashagen, Ulf (Hrsg.), The First Computers - History and Architectures, Cambridge, MA, London 2000, S. 315-322
Petzold, Hartmut, Hardwaretechnologische Alternativen bei Konrad Zuse, erscheint in: Hellige, Hans Dieter (Hrsg.), Geschichten der Informatik. Visionen, Paradigmen und Leitmotive, Berlin, Heidelberg, New York 2003
Peatman, J. B., The Design of Digital Systems, New York u.a. 1972.
Phister, Montgomery, Logical Design of Digital Computers, New York, London, Sydney 1958 (A kyb 327 e/502)
Pierce, William H., Failure-tolerant Computer Design. With a bibliography by Paul A. Jensen, New York u.a. 1965 (A kyb 333/427)
Pollard, L. Howard, Computer Design and Architecture, Englewood Cliffs, N.J. 1990
Prinz, D. G.; Smith, J. B., Machines for the Solution of Logical Problems, in: Bowden, Bertram Vivian, (Hrsg.), Faster Than Thought. A Symposium on Digital Computing Machines, New York, London 1953, S. 181-198
Proceedings First Annual Symposium on Computer Architecture, IEEE, Dezember 1973, New York 1973, in: Computer Architecture News, Bd. 2, Nr. 4
Proceedings Second Annual Symposium on Computer Architecture, IEEE, Dezember 1974, New York 1975, in: Computer Architecture News, Bd. 3, Nr. 4
Proceedings Third Annual Symposium on Computer Architecture, IEEE, Pittsburgh, Penn. Januar 1976, New York 1976, in: Computer Architecture News, Bd. 4, Nr. 4
Proceedings Fourth Annual Symposium on Computer Architecture, IEEE, Januar 1977, New York 1977
Proceedings Fifth Annual Symposium on Computer Architecture, IEEE, 1978, New York 1978
Proceedings Eighth Annual Symposium on Computer Architecture, IEEE, Minneapolis, 1981, New York 1981
Proceedings Tenth Annual International Symposium on Computer Architecture, IEEE, Stockholm, Juni 1983, Los Alamitos, Cal. 1983
Proceedings Eleventh Annual Symposium on Computer Architecture, IEEE, Ann Arbor, Mich. Juni 1984, New York 1984
Proceedings Twelfth Annual Symposium on Computer Architecture, IEEE, 1985, New York 1985
Proceedings Thirteenth Annual Symposium on Computer Architecture, IEEE, Tokyo Mai/Juni 1986, New York 1986

Proceedings Fifteenth Annual Symposium on Computer Architecture, IEEE, Honolulu Mai/Juni 1988, New York 1988

Proceedings Twentieth Annual International Symposium on Computer Architecture, Mai 16 - 19, 1993, San Diego, Cal., Los Alamitos, Cal. u.a. 1993, IEEE Computer Soc. Pr, 1993, San Diego, Calif. 1993

Pugh, Emerson W.; Johnson, Lyle R.; Palmer, John H., IBM's 360 and Early 370 Systems, Cambridge, Mass., London 1991

Randall, Brian, The Origins of Digital Computers. Selected Papers, Berlin, Heidelberg, New York 1973; 3. Aufl. 1982

Rechenberg, Peter, Grundzüge digitaler Rechenautomaten, München 1964; 2. Aufl. München, Wien 1968;

Rechtin, Eberhardt; Maier, Mark W., The Art of Systems Architecting, Boca Raton, Fla. 1997

Recoque, Alice, Survey of Main Trends in Computer Hardware Architecture, in: Information Processing '80, IFIP Proceedings, Amsterdam 1980, S. 115-125

Reddi, S. S.; Feustal, E. A., A Conceptual Framework for Computer Architectures, in: ACM Computing Surveys 8 (1976) 2, S. 277-300

Report on the ENIAC (Electronic Numerical Integrator and Computer, June1, 1946; im Internet unter: http://ftp.arl.army.mil/~mike/comphist/46eniac-report

Rhyne, V. Thomas, Fundamentals of Digital Systems Design, Englewood Cliffs, N. J. 1973

Richter, Reinhard; Sander, Peter; Stucky, Wolffried, Der Rechner als System. Organisation, Daten, Programme, Stuttgart 1997

Rojas, Raúl, Who invented the Computer ? The debate from the viewpoint of computer architecture, in: W. Gautschi (Hrsg.), Fifty Years Mathematics of Computation, Proceedings of Symposia in Applied Mathematics, AMS, 1993, S. 361-366

Rojas, Raúl, On Basic Concepts of Early Computers in Relation to Contemporary Computer Architectures, in: Proceedings of the 13th World Computer Congress, Hamburg, Bd, 2, S. 324-331

Rojas, Raúl, Die Architektur der Rechenmaschinen Z1 und Z3 von Konrad Zuse, in: Informatik-Spektrum 19 (1996), S. 303-315

Rojas, Raúl, Konrad Zuse‘s Legacy: The Architecture of the Z1 and Z3, in: Annals of the History of Computing 19 (1997) 2, S. 5-16

Rojas, Raúl, Konrad Zuses Rechenmaschinen: Sechzig Jahre Computergeschichte, in: Spektrum der Wissenschaft, Mai 1997, S. 54-62

Rojas, Raúl, How to Make Zuse's Z3 a Universal Computer? in: International Conference on the History of Computing, Paderborn (14. - 16. 8. 1998), Preprint der Proceedings, 1998, S. 137-144

Rojas, Raúl (Hrsg.), Die Rechenmaschinen von Konrad Zuse, Berlin, Heidelberg, New York 1998

Rojas, Raúl, Die Architektur der Rechenmaschinen Z1 und Z3, in: Rojas, Raúl (Hrsg.), Die Rechenmaschinen von Konrad Zuse, Berlin, Heidelberg, New York 1998, S. 27-62

Rojas, Raúl, The Architecture of Konrad Zuse’s Early Computing Machines, in: Rojas, Raúl; Hashagen, Ulf (Hrsg.), The First Computers - History and Architectures, Cambridge, MA, London 2000, S. 237-261

Rojas, Raúl; Hashagen, Ulf (Hrsg.), The First Computers - History and Architectures, Cambridge, MA, London 2000

Rosen, Saul, Electronic Computers: A Historical Survey, in: Computing Surveys 1 (1969) 1, S. 7-36

Rosin, R. F., Contemporary Concepts of Microprogramming and Emulation, in: Computing Surveys 1 (1969) 4, S. 197-212

Roth, J. P., Systematic Design of Automata, in: AFIPS Bd. 27 (FJCC) 1965, S.1093-1100

Rutishauser, Heinz; Speiser, Ambros P.; Stiefel, Eduard, Programmgesteuerte digitale Rechengeräte, in: Zeitschrift für angewandte Mathematik und Physik 1 (1950) 5, S. 277-297; 1 (1950) 6, S. 339-362; 2 (1951) 1, S. 1-25; 2 (1951) 2, S. 63-91; wiedergedruckt als: dies., Programmgesteuerte digitale Rechengeräte (elektronische Rechenmaschinen), Mitteilungen aus dem Institut für angewandte Mathmatik an der ETH Zürich, Nr. 2, Basel, Stuttgart 1951

Salisbury, A. B., Microprogrammable Computer Architectures, New York 1976

Samuel, Arthur, L., Computers with European Accents, in: AFIPS 11 (1957) WJCC, S. 14-17

Schecher, H., Funktioneller Aufbau digitaler Rechenanlagen, Berlin, Heidelberg, New York 1973

Schmid, D.; Senger, D.; Wojtkowiak, H, Technische Informatik Teil I: Grundprinzipien des Entwurfs und der Organisation digitaler Rechenanlagen, München 1973

Schmidt-Brücken, Katharina, Die Rechenmaschinenspeicher als Gedächtnis. Einflüsse der Neurophysiologie auf den Rechnerbau in den vierziger Jahren, in: Siefkes, Dirk; Eulenhöfer, Peter; Stach, Heike; Städtler, Klaus (Hrsg.), Sozialgeschichte der Informatik. Kulturelle Praktiken und Orientierungen, Wiesbaden 1998, S. 197-211

Schneck, Paul B., Supercomputer Architecture, Boston, Dordrecht, Lancaster 1987

Schuff, Hans Konrad, Entwicklungstendenzen im Bereich der elektronischen Rechenanlagen, in: Elektronische Datenverarbeitung 1 (1960) 1, S. 1-9

Schunemann, C., Mikro- und Pico-Programmspeicher, in Hasselmeier, H.; Spruth, W. G., Rechnerstrukturen, München, Wien 1974, S. 36-74

Serrell, R.; Astrahan, M. M.; Patterson, G. W.; Pyne, I. B., The Evolution of Computing Machines and Systems, in: Institute of Radio Engineers, Proceedings 50 (Mai 1962), S. 1039-1058

Shannon, Claude E., Computers and Automata (1953), wiedergedruckt in: Pylyshyn (Hrsg.), Perspectives on the Computer Revolution, S. 114-127.

Shaw, Mary; Garlan, David, (Software Architecture. Perspectives on an Emerging Discipline, London 1996

Shiva, Sajjan G., Computer Design and Architecture, 3. Revised and Expanded Edition, Marcel Dekker 2000

Shuey, Richard L.; Spooner, David L.; Frieder., Ophir, The architecture of distributed computer systems : a data engineering perspective on information systems, Reading, Mass. u.a. 1997

Sieworek, Daniel P.; Bell, C. Gordon; Newell, Allen, Computer Structures: Principles and Examples, New York 1982

Sima, Deszö; Fountain, Terence; Kacsak, Péter, Advanced Computer Architectures: A Design Space Approach, Harlow, GB, Reading, MA, Menlo Parc, CA 1997

Simon, Herbert A., Style in Design, in: Eastman, Charles M. (Hrsg.), Spatial Synthesis in Computer Aided Building Design, London 1975, S. 287-309

Simon, Herbert A., The Architecture of Complexity, in: Proceedings of the American Philosophical Society 106 (1962), Dez. S. 467-482; wiedergedruckt in: Simon, Herbert A., The Sciences of the Artificial, Cambridge, Mass., London 1969, S., 2. Aufl. Cambridge, Mass., London 1981, S.

Simon, Herbert A., The Organization of Complex Systems, in: Pattee, Howard H. (Hrsg.), Hierarchy Theory. The Challenge of Complex Systems, New York 1973, S. 1-27

Simon, Herbert A., Die Architektur des Komplexen, in: ders., Die Wissenschaften vom Künstlichen, Berlin 1990, S. 144-172

Simon, Herbert A., Allen Newell: 1927-1992, in: Annals of the History of Computing, 20 (1998) 2, S. 63-76

Skillicorn, D. B., A Taxonomy for Computer Architecture, in: Computer 21 (1988) 11, S. 46-57

Smith, B., The End of Architecture, in: Computer Architecture News 18(1990) 4, S. 10-17

Smith; Richard E., A Historical Overview of Computer Architecture, in: Annals of the History of Computing, 10 (1989).2, S. 277-303

Sobel, Herbert S., Introduction to Digital Computer Design, Reading Mass., Menlo Park, Cal., London 1970 (A kyb 300 eb/470)

Sohi, Gurindar (Hrsg.), 25 years of the International Symposia on Computer Architecture: Selected Papers ACM, New York, NY 1998

Solomon, Martin B., Jr., Economies of Scale and the IBM System /360, in: Communications of the ACM, 9 (1966), 6, S. 435-440

Solvberg; Kung, Information Systems Engineering, Berlin, Heidelberg, New York 1993

Sonnemann, Rolf, Technik und Technikwissenschaften in der Geschichte, in: Wiss. Zs. der TU Dresden, 36 (1987), 4, S. 1-5

Speiser, Ambros P., Das programmgesteuerte Rechengerät an der Eidgenössischen Technischen Hochschule in Zürich, in: Beilage "Technik" aus der Neuen Züricher Zeitung vom 30.8.1950

Speiser, Ambros P., Entwurf eines elektronischen Rechengerätes unter besonderer Berücksichtigung des Erfordernisses eines minimalen Materialaufwandes bei gegebener mathematischer Leistungsfähigkeit (Mitteilungen aus dem Institut für angewandte Mathmatik an der ETH Zürich, Nr. 1), Basel , Stuttgart 1950; 2. Aufl 1954; 3. unveränderte Aufl. 1957

Speiser, Ambros P., Programmgesteuerte digitale Rechengeräte, in: Mitteilung aus dem Institut für angewandte Mathematik an der ETH, Zürich, 2 (1951)

Speiser, Ambros P., Rechengeräte mit linearen Potentiometern, in: Zeitschrift für angewandte Mathematik und Physik 3 (1952), S. S.

Speiser, Ambros P., Stretch - eine neue Entwicklungsstufe der Datenverarbeitungsmaschinen, in: Elektronische Datenverarbeitung 2 (1961), S. 76-81
Speiser, Ambros P., Eingangs- und Ausgangsorgane, sowie Schaltpult der ERMETH, in: Hoffmann, Walter; Walther, Alwin (Hrsg.), Elektronische Rechenmaschinen und Informationsverarbeitung (Nachrichtentechnische Fachberichte, Bd. 4), Braunschweig 1956, S. 87-89
Speiser, Ambros P., Digitale Rechenanlagen, Berlin, Göttingen, Heidelberg 1961; 2. Aufl. 1964, korrigierter Neudruck 1967
Speiser, Ambros P., Stretch - eine neue Entwicklungsstufe der Datenverarbeitungsmaschinen, in: Elektronische Datenverarbeitung 2 (1961), S. 76-81
Speiser, Ambros P., Neue technische Entwicklungen, in: Hoffmann, Walter (Hrsg.), Digitale Informationswandler. Probleme der Informationsverarbeitung in ausgewählten Beiträgen, Braunschweig 1962, S.
Speiser, Ambros P., The Relay Calculator Z4, in: Annals of the History of Computing, 2 (1980).3, S. 242-245
Speiser, Ambros P., Die Z4 an der ETH Zürich. Ein Stück Technik- und Mathematikgeschichte, in: Elemente der Mathematik 36 (1981) 6, S. 145-153
Speiser, Ambros P., The Early Years of the Institute: Aquisition and Operation of the Z4, Planning of the ERMETH. Departement of Computer Science, ETH-Zürich, 1998
Speiser, Ambros P., Konrad Zuse's Z4: Architecture, Programming, and Modifications at the ETH Zurich, in: Rojas,Raúl; Hashagen, Ulf (Hrsg.), The First Computers - History and Architectures, Cambridge, MA, London 2000, S. 263-276
Speiser, Ambros P., Konrad Zuses Z4 und die ERMETH: Ein weltweiter Vergleich, erscheint in: Hellige, Hans Dieter (Hrsg.), Geschichten der Informatik. Visionen, Paradigmen und Leitmotive, Berlin, Heidelberg, New York 2003
Sprague, Richard E., A Western View of Computer History, in: in: Communications of the ACM, 15 (1972) 7, S. 686-692
Spruth, W. G., Einführung, in: Hasselmeier, H.; Spruth, W. G., Rechnerstrukturen. Vorträge des Informatik-Symposiums der IBM Deutschland, Wildbad 1973, München, Wien 1974, S. 9-13
Spruth, W. G., Trends in Computer System Structure and Architecture, in: Samelson, Klaus, ECI Conference 1976. Proceedings of the 1st Conference of the European Cooperation in Informatics, Amsterdam Aug. 1976 (Lecture Notes in Computer Science, Bd. 44), Berlin, Heidelberg, New York 1976, S. 12-32
Spruth, W. G., Interaktive Systeme, Science Research Associates, 1977
Spruth, W. G., The Design of a Microcomputer, Berlin, Heidelberg, New York 1989
Stabler, E. P., System Description Languages, in: IEEE Transactions on Computers, C-19, 19 (1970) 12, S. 1160-1173
Stallings, William, Computer Organization and Architecture. Designing for Performance, 5. Aufl. Upper Saddle River 2000
Stibitz, George R., Computer (1940), in: Randell, Brian (Hrsg.), The Origins of Digital Computers. Selected Papers, Berlin, Heidelberg, New York 1973, S. 241-246; 2. Aufl. 1975; 3. Aufl. 1982, S. 247-252
Stibitz, George R., Introduction into the Course on Electrical Digital Computers, Lecture 1 (8.Juli 1946), Pennsylvania University, Theory and Techniques for Design of Electronic Digital Computers, Bd. 1, S; wiedergedruckt in: Campbell-Kelly, Martin; Williams, Michael R. (Hrsg.), The Moore School Lectures. Theory and Techniques for Design of Electronic Digital Computers (Charles Babbage Institute (Hrsg), Reprint Series for the History of Computing, Bd. 9), London, Los Angeles, San Francisco 1985, S. 5-16
Stibitz, George R., The Organization of Large Scale Calculating Machinery, in: Harvard University Computation Laboratory (Hrsg.), Proceedings of a Symposium on Large Scale Calculating Machinery, Sponsered by the Navy Department Bureau of Ordnance and Harvard University at the Computation Laboratory Jan. 1947, Cambridge, Mass. 1948, S. 91-100; wiedergedruckt in: in: Proceedings of a Symposium on Large Scale Calculating Machinery, in: Charles Babbage Institute (Hrsg.), Reprint Series for the History of Computing Bd. 7 , London, Los Angeles, San Francisco, S. 91-100
Stibitz, George R., Should Automatic Computers be Large or Small, in: Mathematical Tables and other Aids to Computation 2 (1947) 20, Okt., S. ; wiedergedruckt in: Stibitz, George R., Automatic Computing Machinery, in: Annals of the History of Computing 4 (1982) 2, S. 140-142
Stibitz, George R., A Note on "Is" and "Might Be" in Computers, in: Mathematical Tables and other Aids to Computation 4 (1950) 31, Juli, S. ; wiedergedruckt in: Stibitz, George R.,

Automatic Computing Machinery, in: Annals of the History of Computing 4 (1982) 2, S. 142
Stibitz, George R. (as told to Loveday, Evelyn), The Relay Computers At Bell Labs, in: Datamation 13 (1967) 4, S. 35-44; H.5, S.45-49
Stibitz, George R., Early Computers, in: Metropolis, Nicholas C.; Howlett, Jack; Rota, Gian-Carlo (Hrsg.), A History of Computing in the Twentieth Century. A Collection of Essays, New York, London 1980, S. 479-483
Stibitz, George R., Automatic Computing Machinery, in: Annals of the History of Computing 4 (1982) 2, S. 140-142
Stibitz, George R.; Larrivee, Jules A., Mathematics and Computers, New York, Toronto, London 1957
Stone, Harold S. (Hrsg.), Introduction to Computer Architecture, Chicago, Palo Alto, Toronto 1975; 2. Aufl. Chicago, Ill. 1980 (A kyb 326 e 445)
Stone, Harold S., Introduction to Computer Organization and Data Structures, New York, San Francisco, St. Louis 1972 (A kyb 326/486)
Stone, Harold S., High Performance Computer Architecture, Reprint Readings, Mass, 1987 (A kyb 326 f/71)
Stone, Harold, S.; Sieworek, Daniel P., Introduction to Computer Organization and Data Structures: PDP-11 Edition, New York u. a. 1975
Stone, Harold, S.; Thornton, James E., Computer System Organization: Problems of the 1980's, in: IEEE Computer Sept. 1978, S. 20-28
Strachey, Christopher, The Interactions of Software Engineering and Machine Structure, in: J. S. Hugo (Hrsg.), The Fourth Generation, International State of the Art Report, Berkshire 1971, S. 341-356
Stuart-Williams, R., Special-purpose Automatic Computers, in: Bowden, Bertram Vivian, (Hrsg.), Faster Than Thought. A Symposium on Digital Computing Machines, New York, London 1953, S. 199-202 (bes. über Nimrod)
Su, S. Y. H., A Survey of Computer Hardware Description Languages in the USA, in: Computer 7 (1974) 12, S. 45-51
Swartzlander, E. E. (Hrsg.), Computer Design Development. Principle Papers, Rochelle Park, N. York 1976
Symposium on Computer Architecture: Proceedings of the Annual Symposium on Computer Architecture
Tanenbaum, Andrew S., Implications of Strucured Programming for Machine Architectures, in: Communications of the ACM, 21 (1978), 3, S. 237-246
Tanenbaum, Andrew S., Structured Computer Organization, Englewood Cliffs, N. J. 1976; 2. Aufl. 1984; 3. Aufl. 1990; 4. Aufl. 1999
Tanenbaum, Andrew S.; Goodman, James, Computerarchitektur: Strukturen, Konzepte, Grundlagen, München u.a. 1999
Thornton, J. E., Design of a Computer: The Control Data 6600, Glenview, IL 1970
Thornton, James E., The CDC 6600 Project, in: Annals of the History of Computing 2 (1980) 4, S. 338-348
Thurber, Kenneth J., Large Scale Computer Architecture: Parallel and Associative Processors, Rochelle Park, N. J. 1976
Thurber, Kenneth J.; Masson, Gerald M., Distributed-Processor Communication Architecture, Lexington, Mass., Toronto 1979
Thurber, Kenneth J.; Wald, L. D., Associative and Parallel Processors, ACM Computer Survey, Special Issue on Computer Systems Architecture 7 (1975) 4, S. 215-25
Tiberghien, Jacques (Hrsg.), New Computer Architectures, London, Orlando, San Diego 1984,
Tomek, Ivan, The foundations of computer architecture and organization, New York 1990
Townsend, Ralph, Digital Computer Structure and Design, London 1975 (A kyb 327 f/168)
Toy, W.; Zee, B., Computer Hardware/ Software Architecture, Englewood Cliffs, N. J. 1986
Treleaven, Philip C., Decentralized Computer Architecture, in: Tiberghien, Jacques (Hrsg.), New Computer Architectures, London, Orlando, San Diego 1984, S. 1-58
Treleaven, Philip C., Control-Driven, Data-Driven, and Demand-Driven Computer Architecture, in: Parallel Computing 2 (1985), S.
Turing, Alan M., Collected Works of A. M. Turing, initiiert von P. N. Furbank, 4 Bde. Amsterdam, london, New York 1992 (Bd. 1: Pure Mathematics, hrsg. Von J. L. Britton; Mathematical Logic, hrsg. Von R. O. Gandy u. C.E.M. Yates; Mechanical Intelligence, hrsg. Von Darrel C. Ince; Morphogenesis, hrsg. Von P.T. Saunders
Turing, Alan M., On Computable Numbers, With an Application to the Entscheidungsproblem (1937), in: Davis, Martin (Hrsg.), The Undecidable: Basic Papers on Undecidable

Propositions, Unsolvable Problems and Computable Functions, New York 1965, S.116-151, (Nachdruck); Übersetzung in: Dotzler, Bernhard; Kittler, Friedrich (Hrsg.), Intelligence Service, Berlin 1987, S. 17-60

Turing, Alan M., Über berechenbare Zahlen mit einer Anwendung auf das Entscheidungsproblem (1937), in: Dotzler, Bernhard; Kittler, Friedrich (Hrsg.), Alan M. Turing. Intelligence Service. Schriften, Berlin 1987, S. 17-60

Turing, Alan M., Computability and /Definability, in: Journal of Symbolic Logic 2 (1937), S.

Turing, Alan M., Proposal for Development in the Mathematics Division of an Automatic Computing Engine (ACE), presented to the National Physical Laboratory, 1945; wiedergedruckt in: Computer Science 57, National Physical Laboratory, Teddington 1972; wiedergedruckt in: Carpenter, B. E.; Doran, R. W. (Hrsg.), A. M. Turing's ACE Report of 1946 and other Papers (Charles Babbage Institute, Reprint Series for the History of Computing, Bd. 10), London, Los Angeles, San Francisco 1986, S. 20-105

Turing, Alan M., Lecture to the London Mathematical Society on 20 February 1947; wiedergedruckt in: Carpenter, B. E.; Doran, R. W. (Hrsg.), A. M. Turing's ACE Report of 1946 and other Papers (Charles Babbage Institute, Reprint Series for the History of Computing, Bd. 10), London, Los Angeles, San Francisco 1986, S. 106-124

Ungerer, Theo, Innovative Rechnerarchitekturen : Bestandsaufnahme, Trends, Möglichkeiten, Hamburg, New York, St. Louis 1989

Van Dam, Andries, Computer Architecture and Instruction Set Design, in: Proceedings of NCC 1972, S. 519-527, (Nicht in AFIPS) !

Waldschmidt, Klaus (Hrsg.), Parallelrechner: Architekturen, Systeme, Werkzeuge, Stuttgart 1995, (Vorwort von Wolfgang Händler, S. V-IX)

Walther, Alwin, Moderne Rechenanlagen als Muster und als Kernstück einer vollautomatisierten Fabrik, in: Fritz Erler u. a. (Hrsg.), Revolution der Roboter, München 1956, S. 7-64

Ware, W. H., Digital Computer Technology and Design, 2 Bde. New York 1963 (HB=00)

Warfield, J. N., Principles of Logic Design, London 1963

Weber, H., Ein Programmiersystem zur Unterstützung der Rechnerentwicklung, in: Hasselmeier, H.; Spruth, W. G., Rechnerstrukturen. Vorträge des Informatik-Symposiums der IBM Deutschland, Wildbad 1973, München, Wien 1974, S. 372-394

Wegner, Peter, Programming Languages, Information, Structures, and Machine Organization, New York 1968 (A kyb 410 e/613)

Weyh, Ulrich; Schecher, Heinz, Ziffernrechenautomaten. Struktur und Programmierung, München, Wien 1968 (sehr allgemeine Einführung)

Wigington, R. L., A New Concept in Computing, in: Proceedings of the IRE 47 (1959) 4, S. 516-523

Wilkes, Maurice V., Diskussionsbeitrag, in: Report of a Conference on High Speed Automatic Calculating Machines 1949; wiedergedruckt in: The Early British Computer Conferences edited and introduction by Michael R. Williams and Martin Campbell-Kelly Charles Babbage Institute (Hrsg), Reprint Series for the History of Computing, Bd. 14, London, Los Angeles, San Francisco 1989, S. 151-152

Wilkes, Maurice V., The Best Way to Design an Automatic Calculating Machine, in: Manchester University Computer, Inaugurial Conference, Juli 1951, S. 16-18; wiedergedruckt in: The Early British Computer Conferences edited and introduction by Michael R. Williams and Martin Campbell-Kelly Charles Babbage Institute (Hrsg), Reprint Series for the History of Computing, Bd. 14, London, Los Angeles, San Francisco 1989, S. 182-184; und in: Swartzlander, Jr., Earl E. (Hrsg.), Computer Design Development: Principal Papers, Rochelle Park, N. J. 1976, S. 266-270

Wilkes, Maurice V., Calculating Machine Development in Cambridge, in: Bowden, Bertram Vivian (Hrsg.), Faster Than Thought. A Symposium on Digital Computing Machines, New York, London 1953, S. 130-134

Wilkes, Maurice V., The EDSAC, in: International Symposium on Automatic Digital Computation (National Physical Laboratory, 1953), London 1954; wiedergedruckt in: The Early British Computer Conferences edited and introduction by Michael R. Williams and Martin Campbell-Kelly Charles Babbage Institute (Hrsg), Reprint Series for the History of Computing, Bd. 14, London, Los Angeles, San Francisco 1989, S. 229-231

Wilkes, Maurice V., Automatic Digital Computors, London 1956

Wilkes, Maurice V., Micro-programming, in: AFIPS 14 (Dez. 1958) EJCC, S. 18-20

Wilkes, Maurice V., Logical Design of Computers. Introductory Speech , in: Information Processing. Proceedings of the International Conference On Information Processing, Paris 15-20 Juni 1959, München, London 1960, S. 331-333

Wilkes, Maurice V., Computers, Then and Now (1967 Turing Lecture), in: Journal of the ACM 15 (1968) 1, S. 1-7
Wilkes, Maurice V., The Growth of Interest in Microprogramming, in: Computing Surveys 1 (1969) S. 139-145
Wilkes, Maurice V., Historical Perspective-Computer Architecture, in: AFIPS Bd. FJCC 1972, S. 971-976
Wilkes, Maurice V., Early Computer Developments in Cambridge: The EDSAC, in: Radio and Electronic Engineer 45 (1975) 7, S. 332-335
Wilkes, Maurice V., Memoirs of a Computer Pioneer, Cambridge, Mass., London 1985
Wilkes, Maurice V., The Genesis of Microprogramming, in: Annals of the History of Computing 8 (1986) 1, S. 116-118
Wilkes, Maurice V., Computing Perspectives, San Francisco 1995
Wilkes, Maurice V., The Development of the Stored Program Computer, in: ders., Computing Perspectives, San Francisco 1995, S. 21-31
Wilkes, Maurice V.; Renwick, W., The EDSAC, in: Report of a Conference on High Speed Automatic Calculating Machines 1949, S. 9-12; wiedergedruckt in: Randell, Brian (Hrsg.), The Origins of Digital Computers. Selected Papers, Berlin, Heidelberg, New York 1973, S. 389-393; 2. Aufl. 1975; 3. Aufl. 1982, S. 417-421; und in: The Early British Computer Conferences edited and introduction by Michael R. Williams and Martin Campbell-Kelly Charles Babbage Institute (Hrsg), Reprint Series for the History of Computing, Bd. 14, London, Los Angeles, San Francisco 1989, S. 16-20
Wilkes, Maurice V., Historical Perspectives. Computer Architecture, in: AFIPS Bd. 41 (FJCC) 1972, S. 971-976
Wilkinson, Barry, Computer Architecture: Design and Performance, New York u.a. 1991; 2. Aufl. Upper Saddle River, N. J. 1994
Williams, Michael R., A History of Computing Technology, Englewood Cliffs, N. J. 1985
Williams, Michael Roy; Campbell-Kelley, Martin (Hrsg.), The Early British Computer Conferences (Charles Babbage Institute Reprint Series for the History of Computing, Bd. 14), Cambridge, MA 1989
Williams, Rob, Computer Systems Architecture (With CD-ROM), Addiso-Wesley 2001
Williams, Samuel B., Digital Computing Systems, New York 1959 (HB=00)
Willis, Neil, Computer Architecture and Communications, 2. Aufl. Oxford 1993
Wilner, Wayne T., Design of the B1700, in: in: AFIPS Bd. 41 (FJCC) 1972, S.
Wilner, Wayne T., Problem-/Language-oriented Architecture, in: Boulaye, Guy G.; Lewin, Douglas W. (Hrsg.), Computer Architecture. Proceedings of the NATO Advanced Study Institute, St. Raphael, Sept. 1976 (NATO Advanced Study Institutes Series, Serie C, Bd. 32), Dordrecht, Boston 1977, S. 265-293
Witt, Kurt-Ulrich, Elemente des Rechneraufbaus, München, Wien 1995
Wögerbauer, Hans, Die Technik des Konstruierens, 2. Aufl. Berlin 1943
Yourdon, Edward, Design of On-line Computer Systems, Englewood Cliffs, N.J. 1972 (BB Math.-MZH 19h kyb 520.5/545)
Yuen, C. K., Essential Concepts of Computer Architecture, 1989
Zargham, Mehdi R., Computer Architecture. Single and Parallel Systems, Upper Saddle River, N. J. 1996
Zemanek, Heinz, "Mailüfterl", ein dezimaler Volltransistor-Rechenautomat, in: Elektrotechnik und Maschinenbau 75 (1958) 15/16, S. 453-463
Zemanek, Heinz, Some Philosophical Aspects of Information Processing, in: ders. (Hrsg.), The Skyline of Information processing. Proceedings of the Tenth Anniversary Celebration of IFIP, Amsterdam 1972, S. 93-140
Zemanek, Heinz, Entwurf und Verantwortung, in: IBM-Nachrichten 28 (1978) H. 241, S. 173-182
Zemanek, Heinz, Abstract Architecture. General Concepts for System Design, in: Björner, Dines (Hrsg.), Abstract Software Specifications, Proceedings of the Copenhagen Winterschool 1979 (Lecture Notes in Computer Science, Bd. 86), Berlin, Heidelberg, New York 1980, S. 554-563
Zemanek, Heinz, Gedanken zum Systementwurf. Ein Gebäude und Computer generalisierter Architekturbegriff, der auch für Fahrzeuge und Verkehrssysteme nützlich sein könnte, in: Heinz Maier-Leibnitz (Hrsg.), Zeugen des Wissens Mainz 1986, S. 99-125
Zemanek, Heinz, Das geistige Umfeld der Informationstechnik, Berlin, Heidelberg, New York 1992
Zemanek, Heinz, Das "Mailüfterl". Ein österreichischer Aufbruch ins Computerzeitalter, in: Rafeiner, O. (Hrsg.), Patente, Marken, Muster, Märkte. Der gewerbliche Rechtsschutz international, Wien 1993, S. 142-160

Zemanek, Heinz, The Mailüfterl Adventure. The adventure to start and the satisfaction of having completed a pioneer Transistor Computer (1954 to 1958), in: Aspetti storici ed epistemologici dell'Informatica, AICA 94, Palermo Sept. 1994, Atti del congresso Annuale, S. 3-22

Zemanek, Heinz, Konrad Zuse und die Systemarchitektur, das Mailüfterl und der Turmbau zu Babel, erscheint in: Hellige, Hans Dieter (Hrsg.), Geschichten der Informatik. Visionen, Paradigmen und Leitmotive, Berlin, Heidelberg, New York 2003

Zurcher, F., Randell, Brian, Iterative Multi-Level Modelling. A Methodology for Computer System Design, in: Information Processing 68, Proceedings of the IFIP Congress 1968, Amsterdam 1969, S. 867-871

Zuse, Konrad, Zuse-Internet-Archiv (zit.: ZIA). Gesamtausgabe der Schriften Konrad Zuses, hrsg. von Horst Zuse und Raúl Rojas auf der Basis des Zuse-Abschriften-Archivs im Heinz-Nixdorf MuseumsForum Paderborn, früher GMD Darmstadt (zit. HNF); URL: http://www.zib.de/zuse/Inhaltsverzeichnis/Texte

Zuse, Konrad, Die Rechenmaschine des Ingenieurs, 30.1.1936, HNF 009/001; ZIA 0234

Zuse, Konrad, Verfahren zur selbsttätigen Durchführung von Rechnungen mit Hilfe von Rechenmaschinen, Patentanmeldung Z23139, 11.4.1936, HNF 005/021; ZIA 0230

Zuse, Konrad, Einführung in die allgemeine Dyadik, 1938a, HNF 009/004,; ZIA 0237

Zuse, Konrad, Programmspeicherung. Tagebuchnotizen über die Entwicklung von starren Rechenplänen und lebenden Rechenplänen, 1938b, HNF 025/011; ZIA 0417

Zuse, Konrad, Patentanmeldung Z391, 1941, in: Rojas, R.(Hg.), Die Rechenmaschinen von Konrad Zuse, Berlin u.a. 1998, S. 111-193

Zuse, Konrad, Rechenplangesteuerte Rechengeräte für technische und wissenschaftliche Rechnungen, 1943, HNF 009/006, ZIA 0317

Zuse, Konrad, Einige Gesichtspunkte der Entwicklung programmgesteuerter Rechenanlagen in den letzten 20 Jahren, in: Allgemeines Statistisches Archiv (1952) 4

Zuse, Konrad, Der Computer, mein Lebenswerk, 2. Aufl. Berlin, Heidelberg, New York 1984

Zuse-Ingenieurbüro Hopferau (Hrsg.), Zuse-Rechengeräte, Informationsblatt vom Oktober 1947; wiedergedruckt in: Beauclair, Wilfried de, Rechnen mit Maschinen. Eine Bildgeschichte der Rechentechnik, Braunschweig 1968, S. 77-80